UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA EN INFORMÁTICA SUPERIOR

PROYECTO FINAL DE CARRERA

ANÁLISIS DE IMPACTO Y DESARROLLO DE BUENAS PRÁCTICAS DE AUDITORIA EN BASES DE DATOS ORACLE 11G

AUTOR: MARIO DELGADO PICAZO

TUTOR: IGNACIO-J. SANTOS FORNER

MARZO 2009

AGRADECIMIENTOS

Para terminar algo, primero hay que iniciarlo. Cuando una persona empieza a estudiar, nunca se detiene. Es el comenzar el peldaño más duro, y nunca se puede realizar solo. Por ello quiero aprovechar esta página para agradecer su apoyo a todas las personas que me lo han dado.

En primer lugar, debo agradecer a Ignacio Santos su apoyo intelectual y filosófico, sin cuya entrega en estos meses no habría sido posible llegar a buen término.

A mis padres, Mercedes y Basilio, y mis hermanos, Merche y Noel, que han sabido cómo disfrutar de mis días buenos y resistir mis días malos. A mis abuelas Ramona y Manuela, que siempre están pendientes de los éxitos de sus nietos. A quienes no están, mis abuelos Vicente y Basilio, que sin verles en mí no podría haber hecho nunca esta carrera. Al resto de mi familia, con incontables miembros, por darme vuestro ánimo y tener mi foto de la orla en el salón.

A Laura, por estar siempre a mi lado salga el Sol por donde salga, por escucharme, quererme y apoyarme incondicionalmente.

De una forma o de otra, este trabajo también es vuestro.

Finalmente, a todos mis amigos y compañeros de prácticas. Ojalá nos volvamos a encontrar en el futuro.

Si todo parece estar yendo bien, obviamente has pasado algo por alto.

Anónimo

Índice

1. Objetivos

Este proyecto consiste en realizar un análisis y un estudio de los diferentes tipos de auditoría en Oracle 11g, con el objetivo de estudiar las características de las herramientas disponibles, así como su viabilidad. Mediante experimentación, se estudiará el comportamiento y el rendimiento de las diferentes modalidades de auditoría que Oracle 11g ofrece como motor de la base de datos, obteniendo resultados y comparativas que se utilizarán para preparar un manual de buenas prácticas acerca de la auditoría de recolección de datos, así como unas recomendaciones sobre diferentes niveles de auditoría a considerar. Paralelamente se abordarán productos comerciales de auditoria, y posteriormente se establecerá un análisis de la auditoría obtenida.

2. Introducción a la Auditoría

De una forma muy general, la **auditoría** es una actividad o acción (o un grupo de estos) realizadas por uno o varios elementos (humanos, máquinas...) con el objetivo de prevenir, detectar o corregir errores, omisiones o irregularidades que afecten al funcionamiento de algo.

También podemos definirla como la actividad o conjunto de actividades realizadas para determinar, por medio de la investigación, la adecuación de los procedimientos establecidos, instrucciones, especificaciones, codificaciones y estándares u otros requisitos, la adhesión a los mismos y la eficiencia de su implantación

Estas definiciones se pueden aplicar a muchos tipos de auditoría: *auditoría financiera,* como un examen sistemático de los libros y registros de un organismo social; la *auditoría organizativa*, como la identificación de las debilidades y las amenazas a las cuales se enfrenta una institución; la *auditoría administrativa*,

que proporciona una evaluación cuantificada de la eficiencia con la que cada unidad administrativa de la empresa desarrolla las diferentes etapas del proceso administrativo; y así un largo etcétera.

Sin embargo, este trabajo está centrado en la **Auditoría Informática**. La Auditoría Informática es la revisión y la evaluación de los controles, sistemas, procedimientos de informática, de los equipos de cómputo, su utilización, eficiencia y seguridad de la organización que participan en el procesamiento de la información.

El objetivo fundamental de este tipo de auditorías es lograr una utilización más eficiente y segura de la información, el recurso con más valor de una organización. Todo esto conlleva la utilización de un conjunto de técnicas, actividades y procedimientos, destinados a analizar, evaluar, verificar y recomendar en asuntos relativos a la planificación, control, eficacia, seguridad y adecuación del servicio informático en la empresa, por lo que comprende un examen metódico, puntual y discontinuo del servicio informático, con vistas a mejorar en rentabilidad, seguridad y eficacia.

La Auditoría Informática debe comprender no sólo la evaluación de un ordenador, de un sistema o de un procedimiento específico, sino que además tendrá que evaluar los sistemas de información en general desde sus entradas, procedimientos, controles, archivos, seguridad y obtención de información.

La Auditoría Informática es de vital importancia para el buen desempeño de los sistemas de información, ya que proporciona los controles necesarios para que los sistemas sean fiables y para que tengan un buen nivel de seguridad. Además se debe evaluar todo: informática, organización de centros de información, hardware y software.

2.1. Auditoría Informática

A finales del siglo XX, los Sistemas Informáticos se han convertido en las herramientas más útiles para crear uno de los conceptos más necesarios para cualquier organización empresarial: los Sistemas de Información.

La informática hoy en día está aplicada a la gestión de la empresa, y por eso las normas y estándares propiamente informáticos deben estar, por lo tanto, sometidos a los estándares generales o corporativos de la misma. Por ello, las organizaciones informáticas forman parte de lo que se ha denominado “Gestión de la empresa”, o "*Management*". Cabe aclarar que la informática no gestiona propiamente la empresa, sino que ayuda en la toma de decisiones, es decir, no decide por sí misma. Por consiguiente, debido a su importancia en el funcionamiento de una empresa, existe la Auditoría Informática.

Los principales objetivos de la Auditoría Informática son el control de la función informática, el análisis de la eficiencia de los Sistemas Informáticos, la revisión de la gestión de los recursos materiales, humanos e informáticos y la verificación del cumplimiento de la Normativa General de la empresa en este ámbito.

El auditor informático ha de velar por la correcta utilización de los recursos que la empresa pone en juego para disponer de un eficiente y eficaz Sistema de Información. Claro está, que para la realización de una Auditoría Informática eficaz, se debe entender a la empresa como una organización, ya sea una Universidad, un Ministerio, un Hospital, una Sociedad Anónima o una empresa Pública. Todos utilizan la informática para gestionar su lógica de negocio de forma rápida y eficiente con el fin de obtener beneficios, ya sean del tipo económico, social, etc.

2.2. Necesidad de la auditoría

Los Sistemas Informáticos están, o al menos deberían estar, sometidos a un control. He aquí algunas de las principales razones por las que es necesaria la auditoría, el control:

- ➔ Los ordenadores creados para procesar y difundir resultados o información elaborada pueden producir información errónea si dichos datos son, a su vez, erróneos. En este caso interviene la Auditoría Informática de Datos.

- ➔ Las estaciones de trabajo, servidores y en general, los Centros de Procesamiento de Datos (CPDs) se convirtieron en blancos para el espionaje, la delincuencia y el terrorismo. Por otro lado hay que tener en cuenta que entre el 60% y el 80% de los actos de sabotaje son internos. En este caso interviene la Auditoría Informática de Seguridad.

- ➔ El cumplimiento de la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal [3], que "*tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar*".

Estos son solo algunos de los varios inconvenientes que puede presentar un Sistema Informático, por eso existe necesidad de auditoría.

Vamos a centrar las próximas líneas en explicar la segunda razón por la que es necesaria una auditoría. El objetivo es diferenciar *seguridad* de *auditoría*. Pongamos el ejemplo del robo de un dato: un usuario de una base de datos puede visualizar un dato que considera interesante gracias a que tiene permisos: *seguridad*. Gracias a la seguridad, hay usuarios que no tienen permisos para ver

ciertos datos. Sin embargo, un usuario que sí tiene permisos puede usar ese dato para el beneficio personal. La necesidad, en este caso, de la auditoría, está en poder ver qué usuarios accedieron a ese dato en un momento concreto para poder así usarlo para el beneficio personal.

3. Auditoría en Base de Datos

En este punto se va a explicar las diferentes formas de auditoría informática. Por un lado hablaremos sobre la auditoría de cliente y, por otro, sobre la auditoría en el motor de la base de datos, cada una con sus diferentes alternativas y soluciones.

3.1. Auditoría del Cliente

La auditoría de cliente es la que no pertenece al motor de la base de datos. Es aquella cuyo desarrollo es realizado por el programador de la aplicación o de la base de datos, no por el administrador o auditor de la base de datos. Podemos distinguir dos subtipos de auditoría de cliente: auditoría de aplicativo y auditoría de disparadores.

3.1.1. Auditoría de aplicativo

Las instituciones normalmente incluyen en sus gestores de información pequeñas aplicaciones (*Applets*, *CGIs*, *ActiveX*, etc.) que ayudan a gestionar los datos (datos personales, pedidos, pagos online, control de acceso, etc.). Existen otras instituciones que utilizan aplicaciones para realizar una gran variedad de operaciones complejas y que requieren alta seguridad (portales corporativos, banca por Internet o *brokers*, comercio electrónico o *e-commerce*, redes privadas virtuales o *extranets*, etc.) y esto implica la utilización de una compleja aplicación que gestiona todas estas operaciones.

Por ello es necesario un servicio para poder analizar todas estas aplicaciones, de forma independiente y exhaustiva. Este servicio del que hablamos es la **Auditoría de aplicativo**. En este caso, el alcance de la auditoría se delimitará al funcionamiento del día a día de la aplicación, así como en los procesos en los cuales participe. Es realizada por la aplicación y es totalmente externa a la base de datos.

3.1.2. Auditoría de disparadores

Los disparadores son procedimientos que se ejecutan cuando se produce un evento de base de datos entre los que están las operaciones de Manipulación de Datos, de conexión y desconexión de usuarios, de arranque de la base de datos o de fallo. El acto de ejecutar un disparador se conoce como disparo.

Hay diversos usos para los disparadores. Podemos usarlos para el mantenimiento de restricciones de integridad complejas, que no sean posibles con las restricciones declarativas definidas en el momento de crear la tabla, ya que son anomalías semánticas del modelo Entidad/Relación (E/R); o para avisar automáticamente a otros programas de que hay que llevar a cabo una determinada acción cuando se realiza un cambio en una tabla.

Sin embargo, nos vamos a centrar en el uso de los disparadores para auditar la información contenida en una tabla, registrando los cambios realizados, la identidad de quien los llevó a cabo y otros datos de interés como cuándo se realizaron, desde dónde, etc.

La idea está en que cuando un usuario inserta, actualiza o borra datos de una tabla, se ejecute el disparador correspondiente para copiar la información a una tabla diseñada para ello: una tabla de auditoría. Así, el disparador guarda en la tabla de auditoría información que el administrador de la base de datos o el auditor considere necesario: fecha en la que se produjo el borrado, modificación o inserción, usuario que realizó la acción, tabla sobre la que se realizó la acción, dato anterior a la modificación, dato posterior a la modificación, dato que se borró, etc.

La sintaxis completa para crear disparadores es la siguiente [4]:

```
 CREATE [ OR REPLACE ] TRIGGER [esquema.] nombre_disparador
   { disparador_simple_dml
   | disparador_compuesto_dml
   | disparador_no_dml
   }
   [ FOLLOWS [ esquema. ] nombre_disparador_2 [,[ esquema.
   ] nombre_disparador_3... ]
   [ ENABLE | DISABLE ]
   [ WHEN (condicion) ]
   Cuerpo_disparador

 disparador_no_dml := { BEFORE | AFTER }
 { evento_ddl [OR evento_ddl]...
 | evento_bd [OR evento_bd]...
 }
 ON { [esquema.] SCHEMA
    | DATABASE
    }

 Disparador_simple_dml := { BEFORE | AFTER | INSTEAD OF }
clausula_evento_dml
   [ clausula_refrencia ] [ FOR EACH ROW ]

 Disparador_compuesto_dml := FOR clausula_evento_dml
[clausula_refrencia ]

 Clausula_evento_dml := { DELETE | INSERT | UPDATE
      [ OF columna [, columna ]... ]
 }
   [ OR { DELETE | INSERT | UPDATE
        [ OF columna [, columna]... ]
        }
   ]...
 ON { [esquema. ]tabla
    | [ NESTED TABLE nombre_columna OF ]
          [ esquema. ] vista
    }

 Clausula_referencia := REFERENCING
  { OLD [ AS ] anterior
  | NEW [ AS ] nuevo
  | PARENT [ AS ] padre
  }...

 Cuerpo_disparador := { bloque_plsql |
bloque_disparador_compuesto | CALL nombre_rutina }

 bloque_disparador_compuesto := no necesario en auditoría
```

3.1.3. Herramientas comerciales para el desarrollo de disparadores

A continuación se analizarán tres herramientas comerciales que, en distinto modo, permiten la creación de disparadores de una forma visual. Se expondrán ventajas e inconvenientes observados.

3.1.3.1. DB Tools for Oracle, Version 5.0.5.131, de SoftTree Technologies

Esta herramienta contiene un conjunto de programas muy completo para el administrador, auditor y desarrollador de bases de datos. Se trata de un sistema con interfaz gráfico con apariencia Windows, de muy fácil instalación y configuración, que permite visualizar la información recopilada de un modo sencillo y claro.

Permite crear disparadores de una forma muy sencilla. Posee un programa llamado *DB Audit Expert*, entre otros, que se conecta a la base de datos, con rol *SYSDBA* introduciendo la contraseña, y visualiza todas las tablas de todos los usuarios que existen en la base de datos.

Sólo hay que indicar la tabla o tablas que deben ser auditadas, pulsar el botón *"Proceed"*, y la herramienta se encarga de crear el disparador y de activarlo.

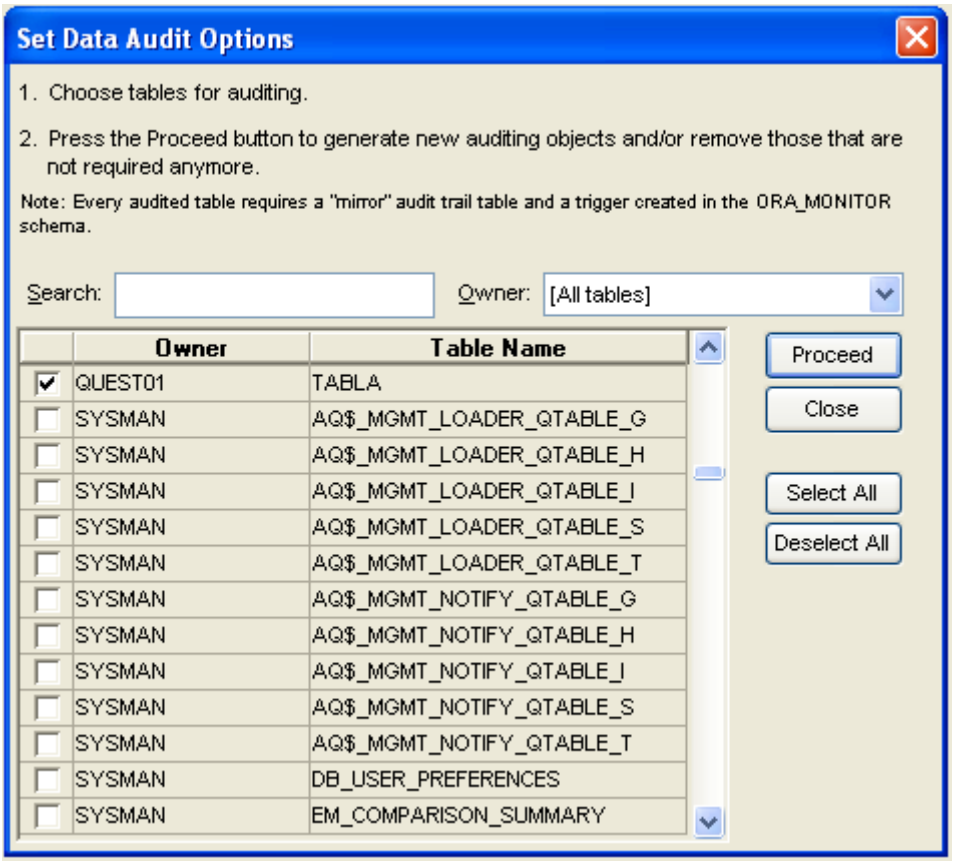

Ilustración 1: creación de auditoría de disparadores en DC Audit Expert

Además, crea una tabla de auditoría ajena a AUD$ de la auditoría genérica (ver punto **3.1 Auditoría genérica**) y de FGA_LOG$ de la auditoría de grano fino (ver punto **3.2 Auditoría de grano fino**) propia con los campos mínimos suficientes para realizar una auditoría: momento en el que se realiza la operación (*timestamp*), usuario que la realizó, ordenador desde el que la realizó, operación ejecutada y todos los valores de todos los campos.

Además tiene herramientas para volcar la información de auditoría en una tabla auxiliar y vaciar la tabla de auditoría con una gran facilidad. Sólo hay que indicar qué tabla es la que el auditor elige cuya información de auditoría quiere que sea volcada en otra tabla y pulsar el botón "*Archive*".

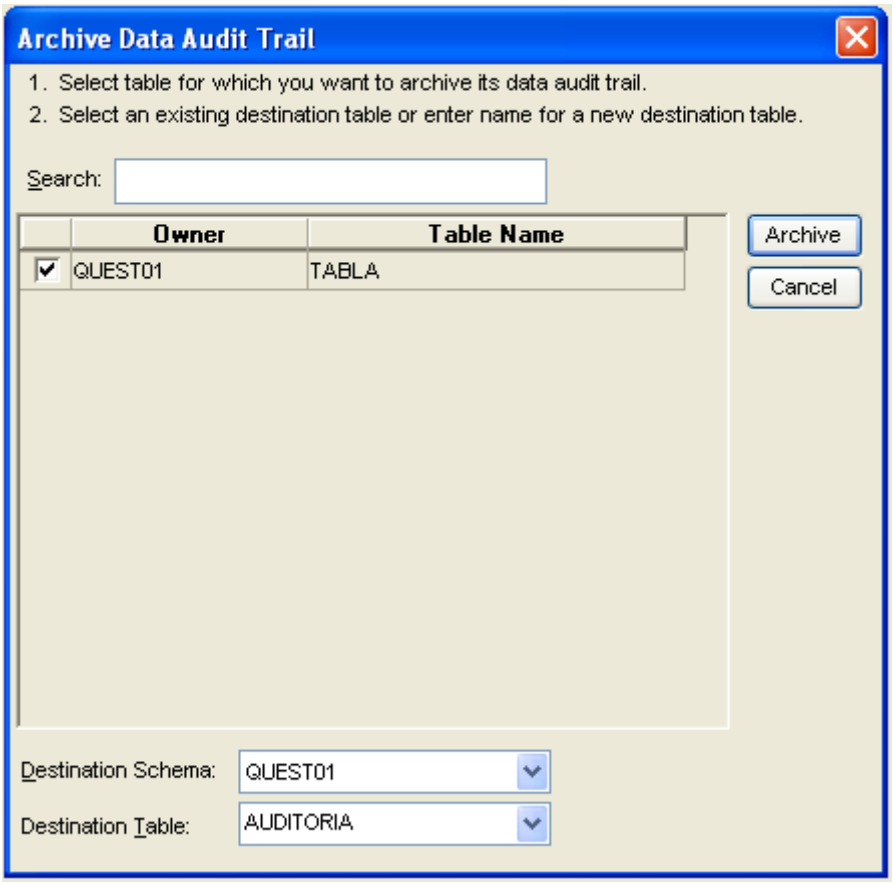

Ilustración 2: Volcado de información de tabla de auditoría de disparadores en DC Audit Expert

El principal problema es que la operación por defecto es la de inserción, y no se puede elegir otra, al menos en esta versión del producto. Esto limita mucho la herramienta, teniendo que hacer, por medio de auditoría *Oracle*, el borrado y modificación (los disparadores no soportan las consultas).

El proceso de crear una auditoría *Oracle* no es más complejo, pero la herramienta no se hace tan rentable como para crear disparadores. La ventaja es que ofrece una interfaz muy sencilla de utilizar, y no es necesario aprenderse toda la sintaxis para realizar cualquier tipo de auditoría. Sin embargo, al crear una auditoría, la interfaz tiene un cuadro *checkbox* para elegir el objeto a auditar, por lo que hace un barrido en toda la base de datos buscando todos los objetos existentes. Esta operación tarda un tiempo relativamente alto comparado con el

tiempo que se tarda en escribir la sentencia de auditoría en SQL, teniendo en cuenta que la sintaxis de la auditoría genérica no es compleja.

Dos inconvenientes a tener en cuenta son, por un lado, que esta herramienta no es compatible con Windows Server, por lo que si el servidor de base de datos es de este tipo, el auditor no podrá usarlo en el mismo equipo, debiendo usar otro ordenador para realizar su trabajo; y por otro lado, no asegura compatibilidad con la versión 11g de *Oracle*. Por ello, aunque al realizar auditoría de disparadores y auditoría genérica no ha dado problema, es posible que en otros programas incluidos en *DB Tools* que no sean *DB Audit Expert* los pueda dar.

3.1.3.2. Oracle Maestro versión 8.12.0.2

Oracle Maestro es una herramienta de gestión para servidores de bases de datos Oracle, una aplicación totalmente visual que dispone de una serie de utilidades diseñadas para realizar operaciones sobre la base de datos.

A diferencia de *Audit Expert* incluido en *DB Tools for Oracle* de *SoftTree Technologies*, *Oracle Maestro* no es una herramienta de auditoría, sino de administración. Sin embargo, permite realizar disparadores para la auditoría de cliente.

Oracle Maestro ofrece opciones interesantes como la optimización de peticiones SQL, la edición de datos BLOBs, la visualización de diagramas ER o la posibilidad de limpiar código PL/SQL.

Brinda soporte para versiones desde la 8i hasta la 11g de Oracle, así como para todas sus características como funciones, procesos, secuencias, clusters, usuarios, vistas, etc. Incluye también un debugger PL/SQL, una herramienta para visualizar gráficamente las peticiones a la base de datos, un visor OLAP, etc.

Crear disparadores con esta herramienta es bastante sencillo. Posee una interfaz fácil de utilizar, y muy eficiente. Sólo hay que elegir la tabla sobre la que

queremos activar el disparador y abrir el editor de tablas sobre ella. En el menú de disparadores (*Triggers*) apretamos el botón derecho del ratón en la lista de disparadores (inicialmente vacía) y elegimos *Crear nuevo disparador…* (*Create New Trigger…*)

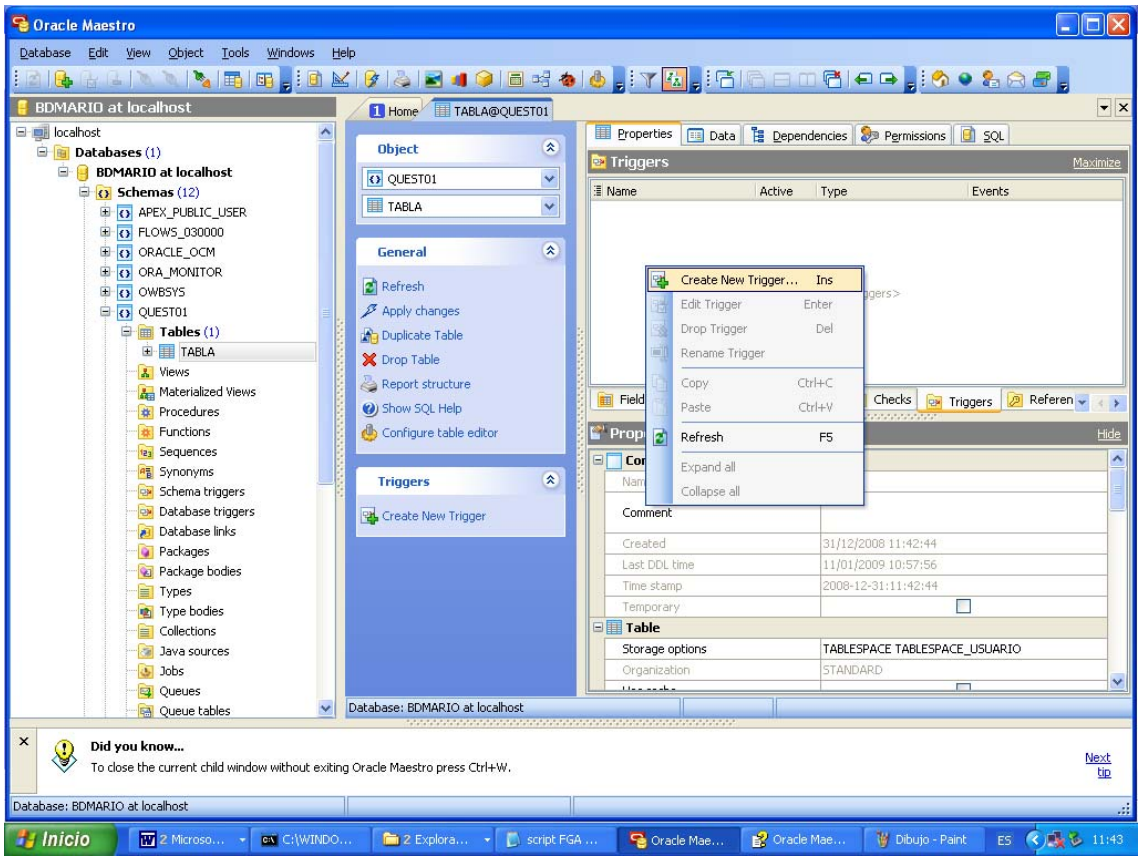

Ilustración 3: Creación de auditoría de disparadores con Oracle Maestro (paso 1)

Se elige el nombre del disparador, la o las operaciones sobre las que se va a activar dicho disparador y el resto de opciones.

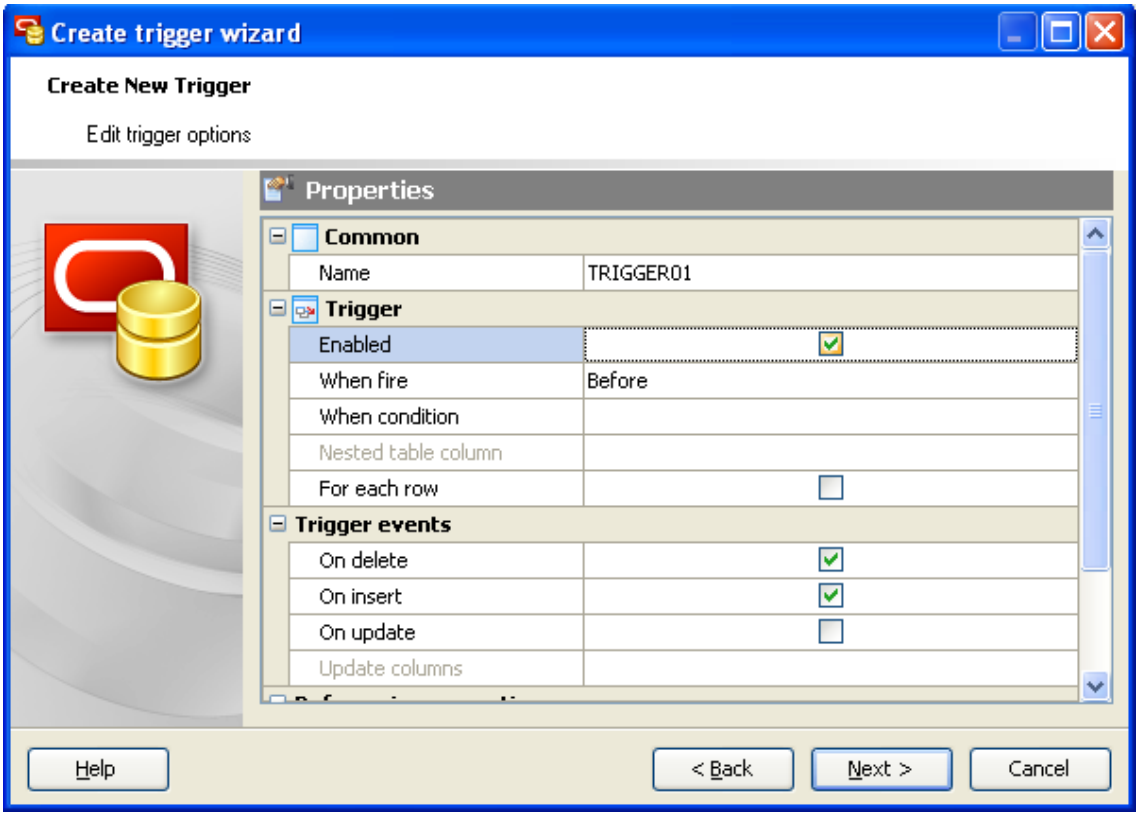

Ilustración 4: Creación de auditoría de disparadores con Oracle Maestro (paso 2)

Y finalmente se escribe el cuerpo del disparador en un editor.

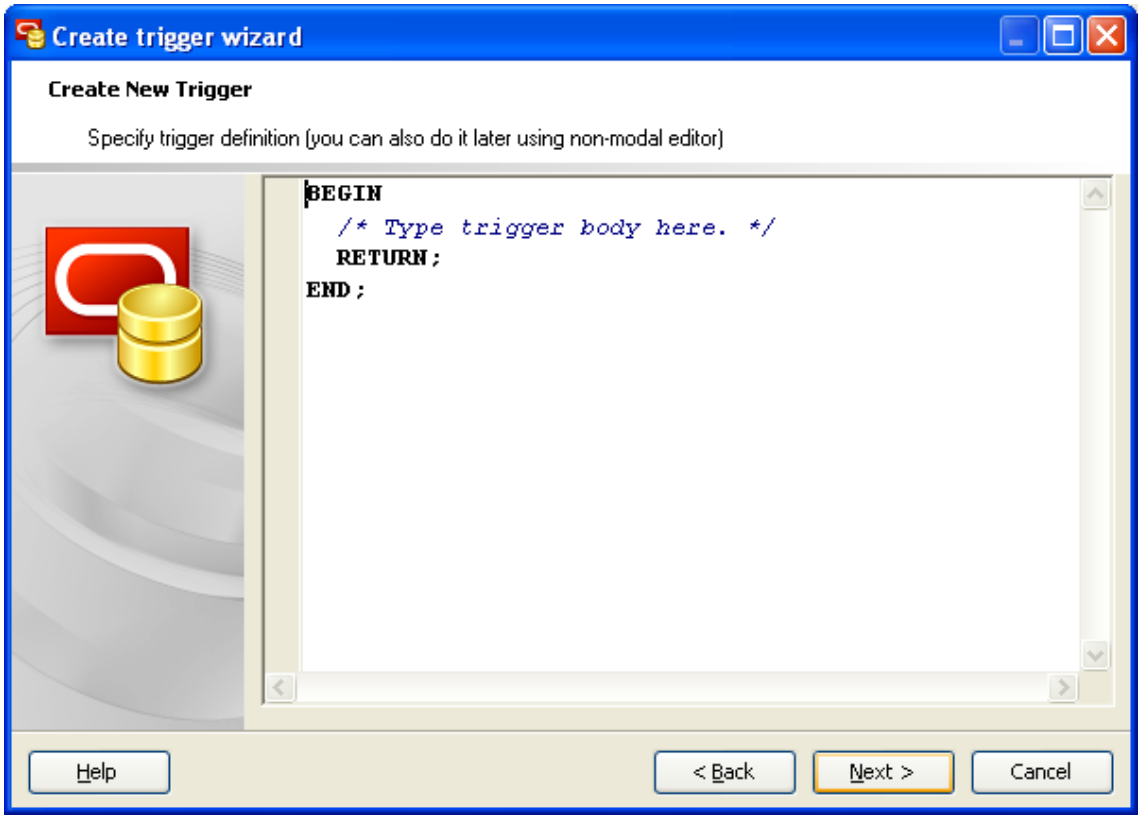

Ilustración 5: Creación de auditoría de disparadores con Oracle Maestro (paso 3)

Desde el punto de vista del auditor, éste es el principal inconveniente de la herramienta comparada con la herramienta anterior *DB Tools for Oracle*. La herramienta anterior permite crear disparadores específicamente para realizar auditoría teniendo un esquema propio automáticamente construido. Es cierto que *Oracle Maestro* ofrece mucha más flexibilidad, pero en definitiva no tiene mucha mejora con respecto a la creación de disparadores por línea de comandos.

Lo mismo pasa con las auditorías. *Oracle Maestro* no facilita la labor de crear auditorías. Para crearlas, se debería ejecutar la sentencia de creación de auditorías *audit*.

3.2. Auditoría en el motor de la base de datos

Oracle ofrece un servicio de auditoría en el motor de la base de datos. Este servicio evita tener que usar los disparadores u otras herramientas externas, con todas sus limitaciones, para auditar la base de datos.

3.2.1. Recolección de datos

La auditoría enfocada desde el punto de vista del motor de la base de datos está basada en políticas de auditoría. Cada política está diseñada de tal forma que sea posible auditar un objeto de la base de datos en unas determinadas circunstancias: bajo el uso de una o varias operaciones definidas por el auditor.

De esta forma, se puede definir la política de auditoría como la información que necesita el motor de la base de datos para auditar. Indica qué hay que auditar, cómo y cuándo. Las diferencias entre el uso de las políticas de auditoría y el uso de disparadores se encuentran en el apartado **3.3 Diferencias entre auditoría en el motor de base de datos y auditoría de disparadores** del presente documento.

La política de auditoría facilita el trabajo al auditor, pues conlleva una mayor facilidad de uso y mucho menor tiempo de desarrollo con respecto a la creación de un disparador para realizar la misma auditoría.

Activada la política, cada vez que se haga una operación sobre el objeto del esquema que satisfaga las circunstancias definidas por el auditor, se creará un registro en una tabla específica indicando todos los parámetros de información necesarios que el motor de la base de datos considera relevantes.

Como veremos en el punto **3 Recolección de datos en Oracle 11g**, hay dos tipos de auditoría: Auditoría genérica y Auditoría de grano fino. Cada tipo de auditoría utiliza una tabla diferente en Oracle 11g: AUD$ y FGA_LOG$ respectivamente.

En los puntos **4.1 Auditoría genérica** y **4.2 Auditoría de grano fino** del presente documento están descritos ambos tipos de auditoría.

3.2.1.1. Recolección de datos en auditoría genérica

Para definir una política de auditoría genérica, es requisito indispensable tener el valor *DB* en el parámetro de la base de datos *audit_trail*. Para comprobarlo, el administrador debe comprobarlo de la siguiente manera:

```
SQL> show parameters audit
NAME                 TYPE      VALUE
-------------------- --------- --------------------------------
audit_file_dest      string    C:\APP\PROYECTO\ADMIN\BDMARIO\ADUMP
audit_sys_operations boolean   FALSE
audit_trail          string    DB
```

El hecho de que el parámetro *audit_trail* esté a *DB* significa que la auditoría en el motor de la base de datos está activada y se pueden crear políticas. En el caso de que estuviera a *NONE*, habría que actualizarlo al valor *DB* mediante la siguiente sentencia:

```
alter system set audit_trail = DB scope = spfile;
```

A continuación paramos y arrancamos la base de datos para que los cambios surtan efecto:

```
Shutdown immediate;
Startup;
```

A partir de ese momento se pueden crear políticas de auditoría genérica. Para crear una política se debe ejecutar el comando `audit`. La sintaxis completa se puede encontrar en la documentación de Oracle en línea [7].

Como en este documento nos centraremos en la auditoría sobre tablas de la base de datos, describiremos con suficiente detalle la sintaxis para dicho propósito. Dicha sintaxis es la siguiente:

AUDIT action on esquema.objeto BY ACCESS|SESSION [WHENEVER [NOT] SUCCESSFUL]

Donde *esquema* es el nombre del esquema al que el objeto pertenece, y *objeto* es el nombre del objeto sobre el que va a actuar la política.

El auditor elegirá *BY ACCESS* cuando la política esté definida para que el motor de la base de datos guarde un registro de auditoría cada vez que se ejecute la operación, caso diferente de cuando el auditor elige *BY SESSION*, que se basa en que se guarde un solo registro si, en una sesión, el usuario ha realizado la operación. Si el auditor no elige ninguna opción, Oracle activa *BY SESSION* por defecto.

Las opciones *WHENEVER SUCCNSSFUL* y *WHENEVER NOT SUCCESSFUL* se diferencian en que la primera está definida para guardar un registro de auditoría cada vez que la operación realizada por el usuario tenga éxito, y la segunda en caso contrario. Si se quiere guardar un registro tanto si se tiene éxito como si no, no se deberá poner este parámetro de la sintaxis.

Por último, el terminal *action* toma el valor de la operación que se quiere auditar, y puede ser cualquiera de los siguientes, o varios separados por coma (“,”):

ALTER, AUDIT, COMMENT, DELETE, EXECUTE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE.

Para desactivar la política de auditoría, la sintaxis es similar. La única diferencia está en que en lugar de la palabra clave *AUDIT* se utilizará la palabra clave *NOAUDIT*.

La tabla AUD$ es la que guarda los datos de auditoría que son generados a partir de la ejecución de la operación que satisface las condiciones de la política definida por el auditor.

En el punto **2.2.2.1 Análisis de la recolección de auditoría genérica** está descrito el uso de esta tabla.

3.2.1.2. Recolección de datos en auditoría de grano fino

Para definir y activar una política de auditoría de grano fino se deben tener privilegios para usar el paquete *DBMS_FGA*, ya que éste es el que posee los procedimientos de seguridad utilizadas en la auditoría de grano fino.

Este paquete tiene 4 procedimientos que van a ser utilizadas para manejar las políticas de grano fino. Por orden alfabético:

- ADD_POLICY: crea una política de auditoría de grano fino. Los parámetros del procedimiento son los siguientes:

```
DBMS_FGA.ADD_POLICY(
 object_schema   VARCHAR2,
 object_name     VARCHAR2,
 policy_name     VARCHAR2,
 audit_condition VARCHAR2,
 audit_column    VARCHAR2,
 handler_schema  VARCHAR2,
 handler_module  VARCHAR2,
 enable          BOOLEAN,
 statement_types VARCHAR2,
 audit_trail     BINARY_INTEGER IN DEFAULT,
 audit_column_opts BINARY_INTEGER IN DEFAULT
);
```

Los más importantes, para nuestro propósito, son los siguientes: *object_schema*, *object_name*, *policy_name*, *enable* y *statement_types*. El primero corresponde al nombre del esquema donde se aloja la tabla que va a ser auditada. El segundo corresponde al nombre de dicha tabla. El tercero es el nombre de la política, que debe ser único, ya que no puede haber dos políticas

con el mismo nombre. El tercero indica si, al crear la política, ésta debe ser habilitada o no. El último indica la operación que va a ser auditada.

Un ejemplo de uso de este procedimiento es el siguiente:

```
begin
DBMS_FGA.ADD_POLICY(
object_schema=>’mario’,
object_name=>’tabla’,
policy_name=>’politica_consulta’,
enable=>FALSE,
statement_types=>’SELECT’);
end;
/
```

- DISABLE_POLICY: deshabilita una política de auditoría. Los parámetros del procedimiento son los siguientes:

```
DBMS_FGA.DISABLE_POLICY(
 object_schema  VARCHAR2,
 object_name    VARCHAR2,
 policy_name    VARCHAR2
);
```

El primer parámetro corresponde al nombre del esquema donde se aloja la tabla que va a ser auditada. El segundo corresponde al nombre de dicha tabla. El tercero es el nombre de la política que el auditor quiere deshabilitar.

Un ejemplo de uso de este procedimiento es el siguiente:

```
begin
DBMS_FGA.DISABLE_POLICY(
object_schema=>'mario',
object_name=>'tabla',
policy_name=>'politica_consulta');
end;
/
```

- DROP POLICY: borra la política de auditoría. Los parámetros del procedimiento son los siguientes:

```
DBMS_FGA.DROP_POLICY(
 object_schema  VARCHAR2,
 object_name    VARCHAR2,
 policy_name    VARCHAR2
);
```

El primer parámetro corresponde al nombre del esquema donde se aloja la tabla que va a ser auditada. El segundo corresponde al nombre de dicha tabla. El tercero es el nombre de la política que el auditor quiere deshabilitar.

Un ejemplo de uso de este procedimiento es el siguiente:

```
begin
DBMS_FGA.DROP_POLICY(
object_schema=>'quest01',
object_name=>'tabla',
policy_name=>'politica_consulta');
end;
/
```

- ENABLE_POLICY: habilita la política de auditoría. Si la política está habilitada, sólo entonces el motor de la base de datos guardará los registros de auditoría en la tabla de auditoría. Los parámetros del procedimiento son los siguientes:

```
DBMS_FGA.ENABLE_POLICY(
 object_schema  VARCHAR2,
 object_name    VARCHAR2,
 policy_name    VARCHAR2,
 enable         BOOLEAN
);
```

El primer parámetro corresponde al nombre del esquema donde se aloja la tabla que va a ser auditada. El segundo corresponde al nombre de dicha tabla. El tercero es el nombre de la política que el auditor quiere deshabilitar. El cuarto es opcional, por defecto tiene valor *VERDADERO* (*TRUE*), e indica si se quiere

habilitar la auditoría. Es algo redundante, pero es posible que permita en un futuro realizar otras operaciones.

Un ejemplo de uso de este procedimiento es el siguiente:

```
begin
DBMS_FGA.ENABLE_POLICY(
object_schema=>'quest01',
object_name=>'tabla',
policy_name=>'politica_consulta');
end;
/
```

3.2.2. Análisis de la recolección

El análisis de la recolección se basa en un conjunto de vistas que actúan sobre la tabla donde se guardan los registros de auditoría. La tabla es AUD$ para la auditoría genérica y FGA_LOG$ para la auditoría de grano fino.

Cada vista ofrece distintos tipos de información de forma más clara para el auditor o administrador de la base de datos.

3.2.2.1. Análisis de la recolección de auditoría genérica

Debido a que la auditoría genérica guarda los registros de auditoría en una tabla llamada AUD$, el motor de base de datos *Oracle* ofrece al auditor un conjunto de vistas que permiten el análisis de los datos de dicha tabla desde diferentes enfoques.

A continuación se explicarán dichas vistas:

- DBA_OBJ_AUDIT_OPTS: describe las políticas de auditoría genérica activadas sobre objetos, incluyendo las opciones de auditoría de todos ellos (operaciones que se auditan sobre los objetos) e información relevante: propietario del objeto, nombre del objeto y tipo de objeto.

- USER_OBJ_AUDIT_OPTS: esta vista no es única ni exclusiva del administrador. Cada usuario tienen en su esquema esta vista, llamada de la misma manera, y que muestra las políticas de auditoría genérica creadas sobre objetos de su esquema. Es similar a DBA_OBJ_AUDIT_OPTS sólo que no lleva la columna del propietario del objeto, pues es implícito que el propietario sea el mismo que la vista.

- DBA_PRIV_AUDIT_OPTS: describe las políticas de auditoría sobre privilegios del sistema que están activas en un momento dado.

- DBA_STMT_AUDIT_OPTS: describe las políticas de auditoría sobre privilegios del sistema que están activas en un momento dado. Se diferencia de DBA_PRIV_AUDIT_OPTS en que, en lugar de mostrar sólo el privilegio que se audita, muestra sus opciones.

- DBA_AUDIT_EXISTS: contiene los registros de los eventos sobre la existencia o no existencia de los objetos, incluyendo en dichos eventos todas las políticas producidas por *audit exists* y *audit not exists*.

- DBA_AUDIT_OBJECT: muestra los registros de auditoría producidos por políticas de auditoría genérica que están relacionadas con objetos (tablas, vistas, índices, secuencias, enlaces, disparadores, espacios de tablas, etc.).

- USER_AUDIT_OBJECT: muestra los registros de auditoría producidos por políticas de auditoría genérica que están relacionadas con objetos (tablas, vistas, índices, secuencias, enlaces, disparadores, espacios de tablas, etc.) y que has sido producidas por el usuario

propietario de la vista. Esta vista no es única, y la tienen todos los usuarios en su esquema.

- DBA_AUDIT_SESSION: muestra los registros de auditoría producidos por políticas de auditoría genérica que están relacionadas con inicio y fin de sesión (CONNECT y DISCONNECT).

- USER_AUDIT_SESSION: muestra los registros de auditoría producidos por políticas de auditoría genérica que están relacionadas con inicio y fin de sesión (CONNECT y DISCONNECT) y que has sido producidas por el usuario propietario de la vista. Esta vista no es única, y la tienen todos los usuarios en su esquema.

- DBA_AUDIT_STATEMENT: muestra los registros de auditoría producidos por políticas de auditoría genérica que están relacionadas con las siguientes operaciones: GRANT, REVOKE, AUDIT, NOAUDIT, and ALTER SYSTEM.

- USER_AUDIT_STATEMENT: muestra los registros de auditoría producidos por políticas de auditoría genérica que están relacionadas con las siguientes operaciones: GRANT, REVOKE, AUDIT, NOAUDIT, y ALTER SYSTEM y que has sido producidas por el usuario propietario de la vista. Esta vista no es única, y la tienen todos los usuarios en su esquema.

- DBA_AUDIT_TRAIL: muestra todos los registros de auditoría genérica.

- USER_AUDIT_TRAIL: muestra todos los registros de auditoría genérica y que has sido producidas por el usuario propietario de la vista. Esta vista no es única, y la tienen todos los usuarios en su esquema.

3.2.2.2. Análisis de la recolección de auditoría de grano fino

Debido a que la auditoría de grano fino guarda los registros de auditoría en una tabla llamada FGA_LOG$, el motor de base de datos *Oracle* ofrece al auditor un conjunto de vistas que permiten el análisis de los datos de dicha tabla desde diferentes enfoques.

A continuación se explicarán dichas vistas:

- ALL_AUDIT_POLICIES: describe todas las políticas de auditoría de grano fino declaradas en el sistema, ya sean habilitadas o no.
- ALL_AUDIT_POLICY_COLUMNS: describe todas las políticas de auditoría de grano fino que estén realizadas para realizar una operación sobre una o varias columna específica de ciertas tablas.
- ALL_DEF_AUDIT_OPTS: contiene las opciones por defecto de auditoría aplicados cuando las políticas sean creadas. Si el auditor no especifica ciertas opciones, dichas opciones se configurarán por defecto según esta tabla.
- AUDIT_ACTIONS: contiene códigos de las acciones que pueden ser auditadas. Es una especie de catálogo para la optimización de la auditoría de grano fino por parte de *Oracle*.
- DBA_AUDIT_POLICIES: muestra exactamente los mismos datos que la vista ALL_AUDIT_POLICIES.
- DBA_COMMON_AUDIT_TRAIL: contiene todos los registros de auditoría, tanto genérica como de grano fino.

- DBA_FGA_AUDIT_TRAIL: muestra todos los registros de auditoría realizados con políticas de auditoría de grano fino.

- STMT_AUDIT_OPTION_MAP: es un mapa de opciones de auditoría que contiene códigos de las opciones que pueden tener las políticas de auditoría. Es una especie de catálogo para la optimización de la auditoría de grano fino por parte de *Oracle*.

- V$XML_AUDIT_TRAIL: contiene todos los registros de auditoría, tanto genérica como de grano fino, auditoría de SYS y registros en XML. Cuando los registros de auditoría se traducen a un formato XML OS, se pueden leer con un editor de texto o a través de esta vista, que contiene información similar a la vista DBA_AUDIT_TRAIL.

3.3. Diferencias entre auditoría en el motor de base de datos y auditoría de disparadores

Existen diferencias entre usar este servicio, llamado a partir de ahora *auditoría estándar*, y usar disparadores.

La auditoría estándar permite auditar operaciones DML y operaciones DDL a todo tipo de objeto de esquema y estructuras. Los disparadores permiten auditar sentencias DML contra tablas y sentencias DDL a nivel de esquema o base de datos. Por ejemplo, un disparador no puede auditar la sentencia de consulta *select*.

Con la auditoría estándar, toda la información de auditoría está guardada de forma centralizada y seleccionada automáticamente. Con los disparadores, el auditor debe elegir qué información guardar y dónde. Esto conlleva cierta dificultad en auditoría de disparadores debido a su desarrollo, pero aporta una gran flexibilidad porque es el auditor quien lo realiza y establece los niveles de

auditoría; por ejemplo cuando el auditor quiere guardar poca información de auditoría por motivos de espacio, o cuando desarrolla un Datawarehouse de auditoría, etc.

Debido a la facilidad de uso de la auditoría estándar, tiene un mejor mantenimiento y es menos complejo en el diseño.

Por su naturaleza, los disparadores no pueden auditar por sesión (clausula BY SESSION): se guarda un solo registro si, en una sesión, el usuario ha realizado la operación a auditar. Sin embargo la auditoría estándar sí lo permite.

Igualmente, con los disparadores no se pueden auditar las conexiones y desconexiones de usuarios a la base de datos; algo que sí está permitido con la auditoría estándar.

El principal inconveniente del uso de los disparadores para realizar auditoría es el hecho de que no puede contener las palabras *commit, savepoint* y *rollback*. Sin realizar un *commit*, un posible fallo en la transacción haría que no se guardara el registro de auditoría. Por ejemplo, si un usuario ha visto un dato, habiendo un disparador que guardase la información de dicho evento en una tabla de auditoría, y falla la transacción por cualquier motivo voluntario o involuntario, no se terminaría de guardar el registro en la tabla de auditoría.

Los disparadores se deberían escribir con sumo cuidado, dado que un error de un disparador detectado en tiempo de ejecución causa el fallo de la instrucción de inserción, borrado o actualización que inició el disparador, en el peor de los casos esto podría dar lugar a una cadena infinita de disparos.

4. Recolección de datos en Oracle 11g

Cuando el Sistema Gestor de Base de Datos Oracle tiene que auditar ciertas acciones, incrementa la cantidad de trabajo que el sistema tiene que realizar. Sin embargo, es posible enfocarla para que solo los eventos que sean interesantes sean capturados. La auditoría mal enfocada afecta significativamente el rendimiento computacional y al espacio de almacenamiento, y por ello, el Auditor de la base de datos debe elegir qué auditar con buen criterio.

Para hacer más o menos personalizada la auditoría, Oracle ofrece dos enfoques de auditoría: **auditoría genérica** y **auditoría de grano fino**. La auditoría genérica fue la primera en implementarse, siendo en la versión 8i; mientras que la auditoría de grano fino fue implementada en la versión 9i para consulta, y en la versión 10g para modificación y borrado.

La auditoría genérica es usada para capturar eventos de usuarios sobre cambios y accesos a información de la base de datos. Puede auditar inserciones, modificaciones, borrados y consultas de dicha información, conexiones y desconexiones a la base de datos, etc. Lo hace siempre de forma general. Sin embargo, a veces es necesario auditar bajo circunstancias específicas. La auditoría de grano fino sí permite esto, además de poder capturar eventos sobre qué se ha visto o qué se ha modificado.

A continuación se explicarán diferentes formas de enfocar la auditoría para conseguir el mejor rendimiento posible.

4.1. Auditoría genérica

La auditoría trabaja guardando y recopilando trazas sobre las acciones de usuarios de la base de datos para poder reconstruir, en un futuro, lo que dichos usuarios han hecho. No tiene que ver con lo que el usuario puede o haya podido hacer, ya que de eso se encargan los Administradores de la base de datos en la concesión de permisos. De hecho, ciertos usuarios pueden tener acceso a información privilegiada gracias a los permisos antes citados; y la auditoría nos permite saber qué usuarios han utilizado esa información.

La **auditoría genérica** engloba toda auditoría que es capaz de controlar comandos SQL, recogiendo todas las operaciones posibles desde el Lenguaje de Definición de Datos al Lenguaje de Manipulación de Datos (sus siglas en inglés, DDL y DML respectivamente).

Dichas operaciones se hacen sobre objetos de la base de datos, y se refiere a operaciones de inserción, borrado, modificación y consulta sobre los datos de dichos objetos, creación y borrado de los objetos, tanto si estas operaciones han tenido éxito como si no. Hay que considerar que las operaciones de consulta, borrado y actualización siempre, tanto si se generan datos en la consulta, se borran datos y se actualizan respectivamente como si no, generarán un dato de auditoria, siempre que está auditado. No ocurre lo mismo con la inserción. Si se quiere auditar las inserciones sin éxito, se debe auditar explícitamente con la opción *whenever not successful*.

También entran las operaciones de entrada en el sistema (o *conexión* a la base de datos), mediante nombre de usuario y contraseña, además de la concesión de permisos.

Por supuesto, no se audita todo automáticamente. Como se ha dicho en puntos anteriores, debe ser el Auditor de la base de datos el que selecciones qué operaciones se deben auditar. Todo depende del objetivo de la auditoría, es decir,

hay que estudiar lo que se debe y lo que no se debe auditar, ventajas e inconvenientes.

Como ejemplo genérico, pondremos una tabla con la siguiente descripción:

```
create table tabla (
 id int primary key,
 dni int,
 num float,
 nombre varchar2(50),
 apellido1 varchar2(50),
 email varchar2(100),
 email2 varchar2(100),
 direccion varchar2(200),
 direccion2 varchar2(200),
 descripcion varchar2(4000)
);
```

Esta tabla tiene datos privados, como el DNI o la dirección, cuya consulta puede ser bastante frecuente, y otros como el campo "*descripción*", que por su longitud simulará datos como el currículum vitae, un expediente o un historial médico, por ejemplo. También hay un campo *num* que simboliza valores como, por ejemplo, el sueldo de una persona.

En esta situación, el auditor debe decidir si auditar o no, y si decide auditar, qué debe auditar. Desde el punto de vista de la auditoría genérica, lo más eficiente sería separar los registros clave en tablas diferentes de la siguiente manera:

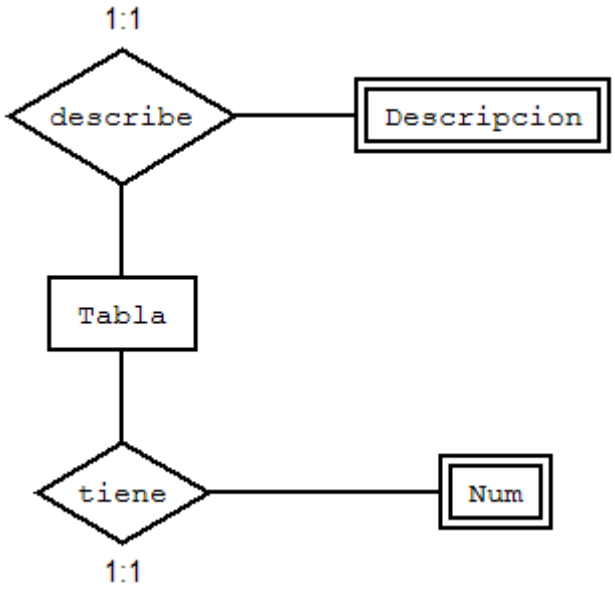

Ilustración 6: Diagrama Entidad/Relación alternativo

Todo ello teniendo que son sólo dos campos los que tienen gran importancia. Campos personales, como el DNI, podrían auditarse, pero auditar ese dato sería auditar todos los datos personales, contenidos en la tabla Tabla, por lo que si se quiere auditar ese dato, se debe auditar la tabla Tabla para mayor eficiencia. Con este esquema, sería muy recomendable auditar las consultas de las tablas Num y Descripción, ya que son los campos más críticos: el sueldo de una persona o su historial médico no tiene por qué consultarse o modificarse a menudo. Por tanto, dependiendo de si la empresa es de gran magnitud o de pequeña magnitud, la auditoría debería ser por sesión (*BY SESSION*) y por acceso (*BY ACCESS*) respectivamente, pues si es de gran magnitud y se realiza auditoría por acceso, se corre el riesgo de que se generen muchos registros de auditoría y se colapse la base de datos en poco tiempo. Con saber que hay alguien que en una sesión ha consultado dicha tabla, sería suficiente.

Sin embargo es muy probable que este cambio no sea rentable, ya que este hecho implicaría un cambio en el modelo relacional, inviable para la mayoría de los casos. Por tanto, habría dos soluciones:

- Usar auditoría genérica para toda la tabla. La ventaja principal sería la sencillez del mantenimiento de esta auditoría.

- Usar auditoría de grano fino. La ventaja sería el menor número de registros de auditoría.

Clave para la elección entre las dos soluciones es el espacio de almacenamiento. Se debería realizar un estudio para ver el porcentaje de consulta del campo a auditar con respecto al número de consultas de la tabla completa.

Midiendo el tamaño máximo de la tupla de cada tabla de auditoría, sale un total de 28996 bytes para la tabla FGA_LOG$ de auditoría de grano fino y un total de 14035 bytes para la tabla AUD$ de auditoría genérica. El cálculo se ha realizado midiendo los registros de tipo CLOB con 4000 caracteres. La razón es de 2,07. La prueba descrita en el punto **5.5 Rendimiento de espacio de tablas** del presente documento se había llegado a un 1,65, sin embargo en esta prueba se utilizaba una consulta de un número mínimo de caracteres (recuerde que de cuanto mayor número de caracteres sea la consulta, mayor espacio ocupará en la tupla de auditoría de grano fino), pondremos una media aritmética entre estas dos medidas: 1.86. Por tanto, si el número de consultas a la tabla incluyendo el campo a auditar es igual o menor que 0.54 veces el número de consultas totales, entonces es más recomendable utilizar auditoría de grano fino que auditoría genérica. Todo ello por temas de ahorro de espacio en disco.

Además, imprescindible sería auditar la modificación en la tabla Num y muy recomendable la tabla Descripción, pues el cambio de sueldo es crítico para una empresa, y la modificación del historial médico o currículum es crítica para la persona.

El borrado y la inserción es menos frecuente en tablas como esta, por lo que no estaría de más auditar el borrado en la tabla Tabla solamente: si se borra o

inserta en la tabla Tabla, se borra o inserta también en el resto de tablas y sólo se crearía un registro de auditoría.

Además, el borrado y la inserción de los datos personales de una persona debe realizarse con el permiso de dicha persona. Por tanto se debe auditar por acceso la inserción y el borrado para comprobar qué ha insertado el usuario de la base de datos. Dado el hipotético caso de que la base de datos tenga datos de una persona sin su permiso (o se hayan borrado sin su permiso), se podrá saber quién insertó (o quién borró) los datos.

4.2. Auditoría de grano fino

En la auditoría genérica, se guarda qué usuarios realizaron qué operación sobre un objeto de la base de datos. Sin embargo, a veces esto no es suficiente. Muchas veces es necesario saber, qué consulta ejecutó un usuario sobre una tabla en un momento determinado o qué datos fueron borrados, modificados o insertados por parte del usuario.

Este tipo de auditoría se llama **Auditoría de grano fino**, y está disponible en Oracle desde su versión 9i. Es capaz de auditar no sólo qué objeto fue consultado por un usuario, sino que también puede auditar qué información obtuvo, en el caso de haber hecho consulta y qué información introdujo, borró o modificó en el caso de haber hecho una modificación. Nótese que la información capturada es muy extensa.

La Auditoría de grano fino surgió por la necesidad de capturar acciones fruto del uso indebido de un privilegio por parte de un usuario [5]. Éste tipo de auditoría podría ser simulada mediante auditoría basada en disparadores unida con el *Log Miner* mejorando el hecho de que, al deshacer la acción mediante la orden *rollback*, se anula la transacción y se elimina la información guardada por el disparador. El *Log Miner* nos permite recuperar la información de los ficheros

de Log, y uniendo sus fuerzas con la Auditoría general, podríamos recopilar la información de las inserciones, borrado o modificaciones.

La clave está en que todas las sentencias SQL ejecutadas por el usuario sobre sus datos, o sobre el diccionario de datos, son grabadas en los archivos Redo Log con el objetivo de que una posible recuperación de la base de datos pueda llevarse a cabo. Con ello es posible realizar una reconstrucción de una tabla en un momento determinado usando dichas sentencias en el mismo orden en que fueron ejecutadas.

Una variante de este tipo de auditoría es guardar información de auditoría sólo cuando cierta columna o columnas de la tabla es o son consultadas. Es muchas veces necesaria tanto para esos casos en los que se hace una consulta de datos sobre dicha columna o sobre datos en los que el campo correspondiente a la columna adquiere un valor específico.

Hay que tener en cuenta que si se audita la inserción, en caso de que ésta no tenga éxito no se guarda el dato de auditoria. Es necesario indicarlo explícitamente usando auditoria genérica.

Pongamos el ejemplo de que en una misma tabla existen datos de personal en la que hay un campo *tipo*, que puede adquirir los valores de '*empleado*' o '*directivo*'. Un empleado que tenga permisos para consultar esa tabla hará ciertas consultas. Sin embargo, al tener mayor importancia para la entidad los datos de los directivos, y debido a que este tipo de auditoría consume muchos recursos de memoria, el Auditor podrá tomar la decisión de auditar sólo las consultas en las que en el resultado aparezca algún directivo.

La Auditoría de grano fino soporta también el número de datos que se observan. Por los mismos motivos que la variante anterior, existe otra a este tipo de auditoría que consiste en auditar aquellas consultas o modificaciones que se hayan hecho sobre un número mínimo de datos. Por ejemplo, el Auditor podrá

realizar auditoría sobre las consultas que, como resultado, hayan dado más de *n* registros.

A todo esto hay que añadir la posibilidad de lanzar un procedimiento almacenado. El procedimiento almacenado está ideado para alertar al administrador/auditor de la base de datos de cualquier anomalía. Otro buen uso que se le puede dar a este procedimiento es el de comprobar cuanto espacio de tablespace queda hasta que se llene, siendo muy importante este hecho, pues si se llena el tablespace en el que se encuentra la tabla de auditoría de grano fino, las futuras operaciones que se vayan a auditar no podrán ser ejecutadas. El procedimiento debería ser creado al menos para la segunda idea para conseguir una alta disponibilidad en la base de datos. Para ello, se indica una referencia a dicho procedimiento en el atributo *handler_module* del procedimiento de creación de la política de auditoría *ADD_POLICY* situado en el paquete *DBMS_FGA* [6].

A continuación se mostrará un ejemplo de procedimiento almacenado que se encarga de, cuando faltan 1500 bytes de espacio libre en el espacio de tablas que contiene la tabla de auditoría de grano fino (TABLESPACE_FGA) envía un correo al auditor de la base de datos, con copia al administrador de la base de datos y copia oculta a un directivo.

```
CREATE OR REPLACE PROCEDURE alerta_espacio AS
tamano REAL;
BEGIN

SELECT ROUND(sum(bytes)/1024,0) INTO tamano FROM
dba_free_space WHERE tablespace_name = 'TABLESPACE_FGA'
GROUP BY tablespace_name;
IF tamano<1500 THEN
  ENVIAR_CORREO('fga@compania.com','auditor@compania.com',
  'administrador@compania.com','gerente@compania.com','Tam
  año TABLESPACEFGA','El tamaño del espacio de tablas que
  contiene la tabla de auditoría de grano fino es inferior
  a 1500Kb.');
END IF;
```

```
END alerta_espacio;
/
```

Para ello, se ha utilizado otro procedimiento para el envío de correos, que se muestra a continuación:

```
 CREATE OR REPLACE PROCEDURE ENVIAR_CORREO (P_SENDER in
varchar2,
 P_RECIPIENT in varchar2,
 P_CC in varchar2,
 P_BCC in varchar2,
 P_SUBJECT in varchar2,
 P_MESSAGE in varchar2) is
 mailhost varchar2(30) := '127.0.0.1';
 mail_conn utl_smtp.connection;
 crlf varchar2(2) := CHR(13)||CHR(10);
 mesg varchar2(4000);
 BEGIN
   mail_conn := utl_smtp.open_connection(mailhost,25);
   mesg := 'Date: '||to_char(sysdate,'dd Mon yy hh24:mi:ss'
   )||crlf||
   'FROM: '||P_SENDER||'>'||crlf||'Subject:
'||P_SUBJECT||crlf||
   'To: '||P_RECIPIENT||crlf||
   'Cc: '||P_CC||crlf||
   'Bcc: '||P_Bcc||crlf||crlf||P_MESSAGE;
   utl_smtp.helo(mail_conn,mailhost);
   utl_smtp.mail(mail_conn,P_SENDER);
   utl_smtp.rcpt(mail_conn,P_RECIPIENT);
   utl_smtp.data(mail_conn,mesg);
   utl_smtp.quit(mail_conn);
 END send_mail;
 /
```

Como ejemplo genérico, volveremos a poner la tabla usada en el punto **3.1 Auditoría genérica** con la siguiente descripción:

```
 create table tabla (
  id int primary key,
  dni int,
  num float,
  nombre varchar2(50),
  apellido1 varchar2(50),
  email varchar2(100),
  email2 varchar2(100),
  direccion varchar2(200),
  direccion2 varchar2(200),
  descripcion varchar2(4000)
```

```
);
```

En este caso no hace falta realizar una reestructuración de la tabla, pues la auditoría de grano fino nos permite auditar por campo.

En primer lugar, habría que auditar las consultas de, como mínimo, los campos críticos: DNI, num o descripción. Con esto el auditor ahorrará la reestructuración de la tabla y las consecuencias de rendimiento.

En cuanto a la modificación, se debería hacer del sueldo, campo crítico de la tabla. Además de ello, la auditoría del sueldo debería hacerse añadiendo un procedimiento que compruebe cuántas veces ha sido modificado el sueldo de esa persona en los últimos *n* días, siendo *n* un número cuya decisión es del auditor, y otro que compruebe si la cantidad de diferencia entre antes y después del cambio sea mayor de *m*. El primer procedimiento debe avisar al auditor en el caso de que se haya cambiado el sueldo a esa persona más de *n* veces, y el segundo deberá avisar al auditor en el caso de que la cantidad de diferencia entre antes y después sea mayor que *m*.

Para guardar la modificación del campo *descrpción*, sería más eficiente hacerlo mediante un disparador, pues es un campo de 4000 caracteres, y el volumen del tablespace que contiene la tabla de auditoría FGA_LOG$ depende de la longitud de la sentencia SQL. Para ello sería más eficiente utilizar un disparador que guarde no sólo qué operación se ha realizado, sino el valor del campo *descripción* y el valor del campo *id* para su localización.

Con respecto a las inserciones y borrados, se debería auditar cada vez que un usuario borra un registro. Con las inserciones ocurre lo mismo que con la modificación del campo *descripción*. Las inserciones no serían recomendables, pues el volumen del tablespace que contiene la tabla de auditoría FGA_LOG$ depende de la longitud de la sentencia SQL. Teniendo un campo de 4000 caracteres, no sería óptimo utilizar la auditoría de grano fino para la inserción.

Para ello sería recomendable utilizar la auditoría genérica o, más eficiente aún, utilizar un disparador que guarde no sólo qué operación se ha realizado, sino qué registro se ha insertado, guardando sólo el campo *id*, identificador del registro insertado.

Como nota importante sobre la auditoría de grano fino hay que decir que, si la inserción está auditada por este método, se debe tener en cuenta que, en el caso de que una inserción falle, no se guarda registro de auditoría. No ocurre lo mismo con el borrado, modificación o consulta: tanto si el borrado o la modificación no se efectúa como si la consulta devuelve un resultado vacío, el registro de auditoría de grano fino sí se crea.

4.3. Herramientas externas a Oracle

En este punto se describirán herramientas externas a *Oracle* que han sido utilizadas en la realización de este documento. Primero se enunciarán herramientas para la medición del rendimiento en las pruebas realizadas y a continuación se enunciarán herramientas de auditoría.

4.3.1. Herramientas de medición de rendimiento

Las siguientes herramientas son las candidatas para la medición del rendimiento de un servidor que tiene una base de datos *Oracle* 11g dadas las pruebas realizadas y descritas en el punto **5 Rendimiento en la recolección de datos en *Oracle* 11g** del presente documento.

4.3.1.1. Monitor de Rendimiento de Windows

El Monitor de Rendimiento (o "*Performance Monitor*") de Windows nos permite supervisar el uso de los recursos de un ordenador, tales como el porcentaje de tiempo del procesador que está trabajando, paginación, interrupciones, memoria disponible, errores de caché, etc. que podemos usar para conocer la carga de trabajo y el efecto que produce en los recursos del sistema,

observar los cambios y las tendencias en las cargas de trabajo y en el uso de los recursos, probar los cambios de configuración u otros trabajos de ajuste mediante la supervisión de los resultados, etc.

Eligiendo los datos más convenientes, podremos medir el rendimiento del ordenador sobre diferentes experimentos que hagamos en este proyecto.

Las herramientas Monitor de sistema y Registros y alertas de rendimiento proporcionan datos detallados acerca de los recursos utilizados por componentes específicos del sistema operativo y por programas que han sido diseñados para reunir información de rendimiento. De hecho, Oracle proporciona sus propios contadores de registros.

Esta herramienta se encuentra en el Panel de Control de Windows → Herramientas Administrativas → Rendimiento.

4.3.1.2. Performance Analysis de Quest

Performance Analysis de Quest es una herramienta utilizada para el análisis de la carga de trabajo de la base de datos Oracle que ofrece una monitorización del sistema, tanto del gestor de base de datos como del propio sistema, y tanto en tiempo real como histórico; siendo utilizado para el seguimiento, análisis y resolución de problemas.

Este componente incrementa la eficiencia del DBA al anticipar problemas de desempeño y facilitar el tiempo para su resolución.

Performance Analysis provee un repositorio central en donde los usuarios pueden recuperar reportes automatizados de estadísticas relacionadas con el desempeño y que ofrece resaltes de actividad a la medida para desglosar a problemas relacionado de forma efectiva.

Tiene el mismo objetivo que Performance Monitor de Windows, pero con una más amplia visión. Es de pago, pero se ha conseguido una licencia de 30 días para la realización de las pruebas. El programa con la licencia de 30 días se puede descargar de la Web de Quest.

Para realizar las pruebas ha sido necesaria una infraestructura de dos ordenadores, llamados TELEMACO y TAIS, tal y como se explica en la siguiente figura:

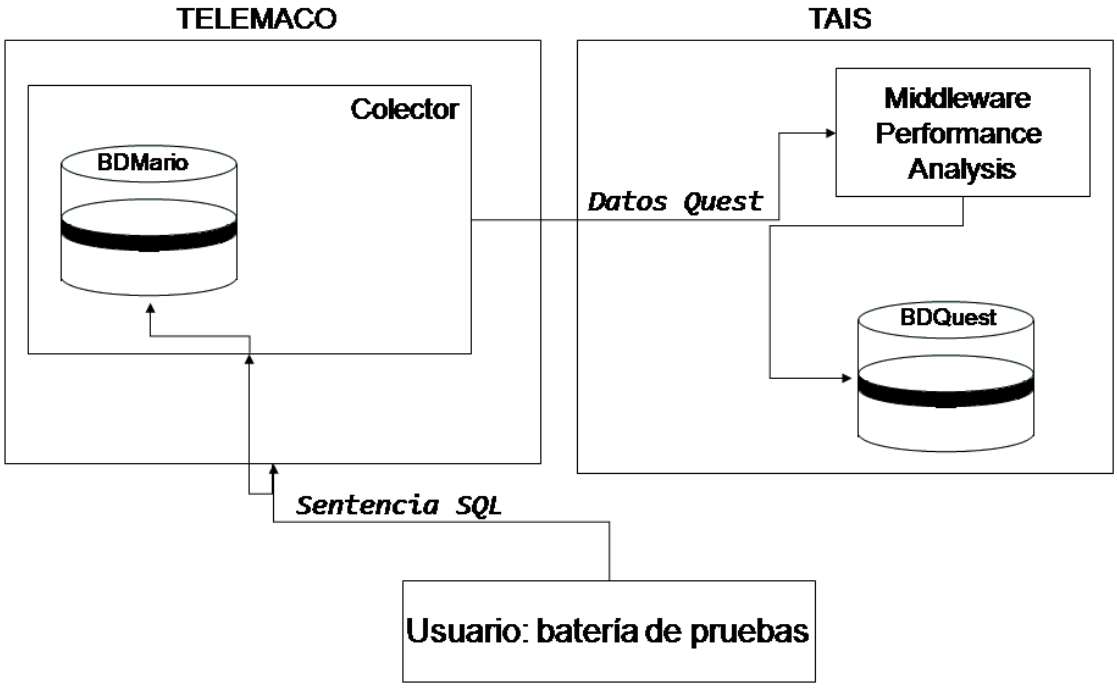

Ilustración 7: Arquitectura de medición de rendimiento con Performance Analysis

El ordenador TELEMACO tendrá el sistema y la base de datos a monitorizar. Posee un proceso de bajo consumo de recursos llamado *Collector* que captura las medidas de rendimiento desde diferentes puntos de vista (porcentaje de tiempo del procesador, memoria RAM utilizada, disco utilizado, etc.) y envía los datos al ordenador TAIS.

El ordenador TAIS tiene un proceso llamado *Middleware Performance Analysis* y una instancia de base de datos *Oracle*. Dicha instancia tiene un

esquema utilizado por *Performance Analysis* para guardar los datos de rendimiento y así tener un histórico. El proceso *Middleware* recibe los datos de la red enviados por el proceso *Collector* del ordenador TELEMACO y los guarda en el esquema anteriormente citado.

El banco de pruebas realizadas descritas en el punto **5 Rendimiento en la recolección de datos en Oracle 11g** del presente documento está hecho con esta herramienta.

4.3.2. Herramientas de auditoría

Las siguientes herramientas han sido estudiadas como herramientas de auditoría. Permiten una mejora en la auditoría de una base de datos *Oracle* y un mayor rendimiento del servidor de base de datos.

4.3.2.1. InTrust de Quest

InTrust es una herramienta que ayuda a recolectar, almacenar, reportar y alertar sobre eventos de datos heterogéneos para resolver las necesidades de las políticas internas, regulaciones externas y mejores prácticas de seguridad.

Permite auditar, obtener informes y generar alertas de actividad de todos los controladores de dominio Windows además de hacer un seguimiento de todos los cambios realizados en el Directorio Activo o en las Políticas de Grupo.

También proporciona una recopilación y almacenamiento de la información de auditoría, permitiendo reaccionar y prevenir posibles violaciones de políticas de seguridad o cambios críticos en objetos del Directorio Activo.

Intrust está compuesto de un conjunto de aplicaciones, cada una con un objetivo diferente:

- Consola de Usuario (*Itrust for Databases User Console*): monitoriza el sistema, detectando accesos sospechosos, generando informes y alertas en tiempo real. Permite crear políticas de auditoría y el acceso a los registros de auditoría.

- Consola Federada (*Intrust for Databases Federated Console*): permite conectarse a servidores que tengan instancias de bases de datos y comparar datos de auditoría.

- Consola de Administrador (*Intrust for Databases Administrator Console*): permite al usuario con rol de administrador (ADMIN) gestionar los servidores de Intrust y los agentes recolectores.

- Constructor de informes (*Intrust for Databases Report Builder*): permite gestionar los informes. Tiene una herramienta de diseño de informes de fácil manejo.

La infraestructura que debe seguir la empresa para el funcionamiento de *Intrust* es la siguiente:

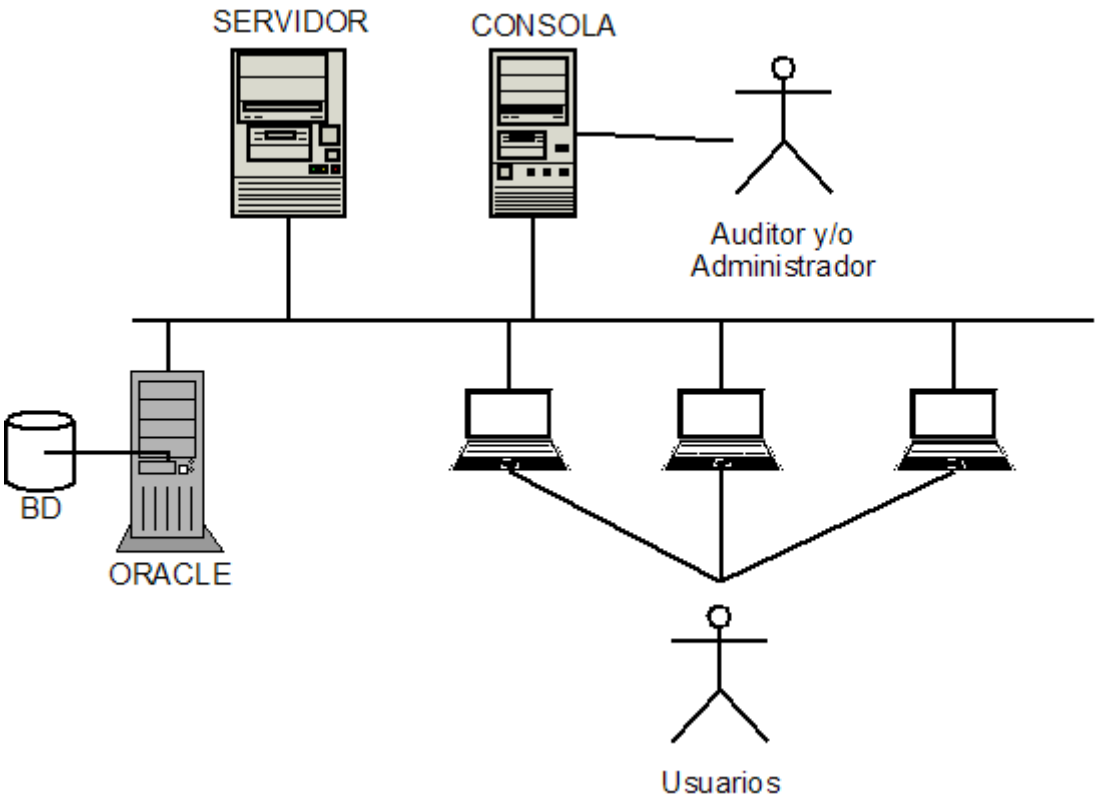

Ilustración 8: Arquitectura de alto nivel del sistema InTrust

Hay tres equipos principales:

- ORACLE: contiene la base de datos a monitorizar.
- SERVIDOR: es una máquina que contiene perfiles, políticas y datos de auditoría. Es el propio almacén de datos de auditoría para *Intrust*. Puede tener sistema operativo Windows o Linux.
- CONSOLA: contiene las cuatro aplicaciones que compone *Intrust*. Recupera la información del servidor para que las aplicaciones puedan realizar las funciones para las que están destinadas cada uno. Sólo puede tener sistema operativo Windows.

La comunicación de los equipos es jerárquica, y se muestra en la siguiente figura:

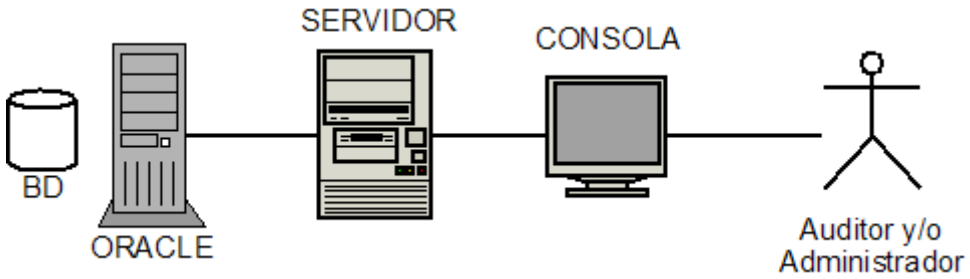

Ilustración 9: Jerarquía de flujo de información en el sistema InTrust

Sobre la estructura el diagrama Entidad/Relación del esquema donde el servidor de *Intrust* guarda la información no hay información alguna.

[14]

4.3.2.2. Oracle Audit Vault

Oracle Audit Vault es una herramienta que permite automatizar la recolección de datos de auditoría, monitorizar y generar informes, volcando dicha información en un esquema propio.

Básicamente, recolecta datos de auditoría de una base de datos, ya sea *Oracle* o MS SQLServer, y, además de guardar los datos en un esquema propio en forma de DataWarehouse, estudia dichos datos para detectar cualquier anomalía.

Oracle Audit Vault consta de un servidor y un agente. El servidor, que en la versión más reciente de *Audit Vault* no está desarrollado para usarlo en entorno Windows, debe ser configurado por el administrador, y se encargará de recolectar todos los datos de auditoría que genera la base de datos, organizarla y generar los informes. El agente se encarga de ofrecer sus servicios al auditor de la base de datos. Desde el agente, el auditor podrá crear políticas de auditoría (sólo para bases de datos Oracle), crear alertas y obtener informes. El agente puede estar

instalado en la misma máquina donde está la base de datos *Oracle* o en otra máquina.

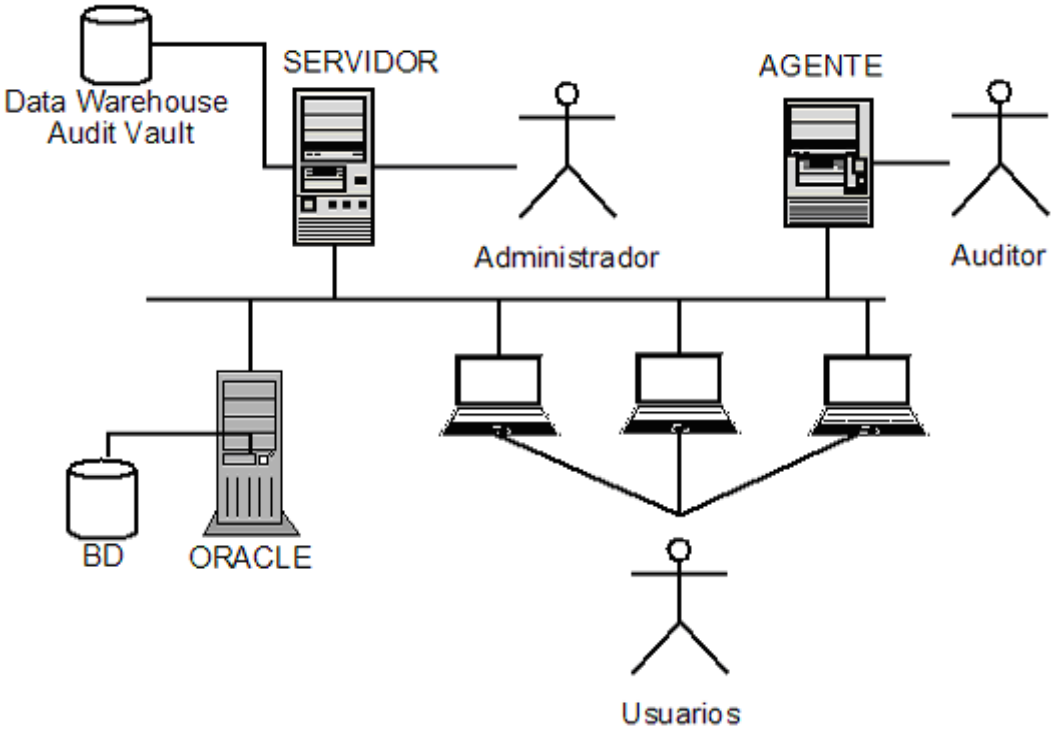

Ilustración 10: Arquitectura de alto nivel del sistema Audit Vault

El flujo de trabajo de *Oracle Audit Vault* se refleja en el siguiente esquema:

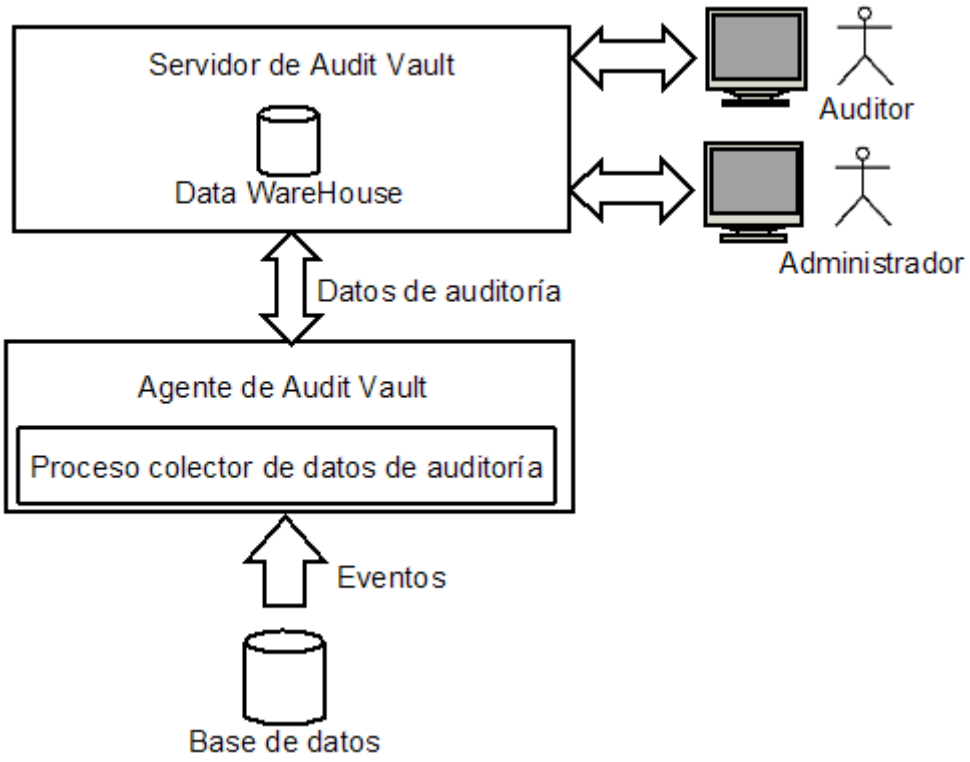

Ilustración 11: Jerarquía del flujo de información del sistema Audit Vault

Las políticas y las alertas detectan amenazas de seguridad estudiando irregularidades en el uso de la base de datos.

Para que todo ello funcione son necesarios al menos dos usuarios: administrador y auditor. El administrador es el propietario de *Audit Vault*. Gestiona los roles y la configuración. El auditor es el que se encarga de gestionar los informes y acceder a los servicios de análisis. Se preocupa de detectar riesgos de seguridad, creando alertas y políticas.

Oracle Audit Vault usa un Data WareHouse modelado por un esquema en forma de estrella. El registro de audtoría está en una tabla (AUDIT_EVENT_FACT) justo en el centro de la estrella, y está descrito por atributos en el resto de tablas. Un esquema en estrella optimiza el rendimiento mediante ejecución de sentencias SQL que aseguran una rápida respuesta.

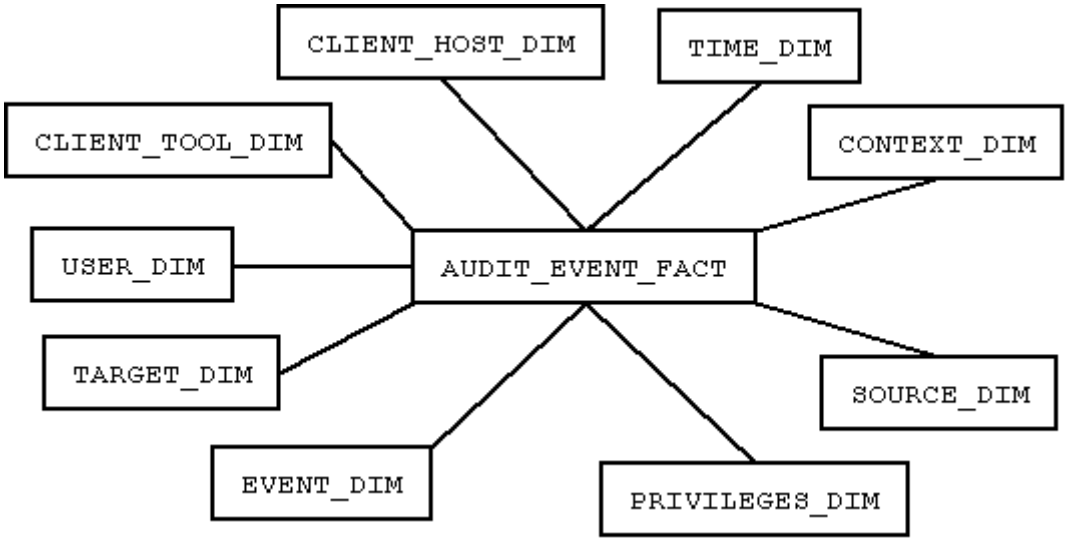

Ilustración 12: Esquema del Data WareHouse que guarda los datos de auditoría en Audit Vault

La tabla AUDIT_EVENT_FACT está en el centro de esta configuración en estrella. Desde ahí, se comunica con el resto de tablas: CLIENT_HOST_DIM, CLIENT_TOOL_DIM, USER_DIM, TARGET_DIM, EVENT_DIM, TIME_DIM, CONTEXT_DIM, SOURCE_DIM, y PRIVILEGES_DIM.

La tabla AUDIT_EVENT_FACT está enlazada a cada tabla por su clave ajena, y contiene el identificador del registro de auditoría, algunos atributos para la generación de informes (reportes) y las claves ajenas del resto de tablas.

- CLIENT_HOST_DIM: contiene información sobre los equipos usados pos los clientes de la base de datos para realizar las operaciones contra ésta.
- CLIENT_TOOL_DIM: contiene información sobre las herramientas utilizadas para conectarse a la fuente de datos de auditoría.
- CONTEXT_DIM: contiene información de contexto de un evento auditado.

- EVENT_DIM: contiene información sobre los eventos que pueden ser ejecutados.

- PRIVILEGES_DIM: contiene información sobre los privilegios usados durante la ejecución del evento.

- SOURCE_DIM: contiene información sobre la base de datos fuente que envió el registro de auditoría al Data WareHouse.

- TARGET_DIM: contiene información sobre el esquema y los objetos del esquema donde los eventos que deben ser auditados so ejecutados.

- TIME_DIM: contiene información sobre el momento en que se ejecutan los eventos. Esta tabla es la más utilizada. Implementa cuatro niveles de jerarquía: día, mes, trimestre y año.

- USER_DIM: contiene información sobre el usuario que ejecutó el evento y al que está asociado.

El principal inconveniente es el hecho de que el servidor de *Audit Vault* no esté disponible para la plataforma Windows. Lo que resuelve este inconveniente es la posibilidad de descarga libre de un sistema operativo Linux compatible desde la página de *Oracle* [8].

Otro inconveniente a priori es el coste de licencia. La licencia para el Servidor de *Oracle Audit Vault* cuesta 50.000 dólares por procesador, y la licencia para el Agente de *Oracle Audit Vault*, 3.000 dólares por procesador [9] (dato del 1 de Enero del 2008). La inversión, para una empresa mediana, puede ser encarecida, pero esa inversión se recupera a medio plazo debido al ahorro en coste de desarrollo y mantenimiento de una infraestructura de auditoría manual. Sin embargo, es una de las ventajas para las grandes empresas.

Otra de las ventajas es la posibilidad de automatizar la instalación. *Oracle Audit Vault* ubica los datos de auditoría en un repositorio distinto al de la base de datos que se quiere auditar. Si esto fuera manual, habría que crear un conjunto de usuarios, enlaces y tablas en forma de base de datos distribuida. Gracias a la automatización, se puede crear esa base de datos distribuida de forma más eficiente.

La tercera ventaja es el uso de las alertas. Se pueden crear alertas fácilmente de forma visual, sin tener que recurrir al desarrollo manual de trabajos ni procedimientos almacenados.

[15]

5. Rendimiento en la recolección de datos en Oracle 11g

En este punto se describirá la realización de un conjunto de pruebas realizadas a una hipotética base de datos sobre *Oracle* 11g, obteniendo diferencias de rendimiento en diferentes situaciones: cuando no se audita nada, cuando se realiza auditoría genérica, cuando se realiza auditoría de grano fino y cuando se realizan las dos a la vez.

También se verá la diferencia del volumen de disco utilizado en cada caso. Todo ello para que, en los puntos **5 Análisis de la recolección** y **6 Normas para auditar elementos de una organización**, se propongan mejoras lo más efectivas posibles.

5.1. Pruebas realizadas e infraestructura

Con el objetivo de medir el rendimiento de las diferentes auditorías, se ha creado una pequeña infraestructura de dos ordenadores en la que cada uno tendrá un rol diferente: uno será el servidor de la base de datos y el otro será el cliente.

En la siguiente figura se muestra un esquema aclaratorio donde se reproduce la situación de las pruebas pertinentes.

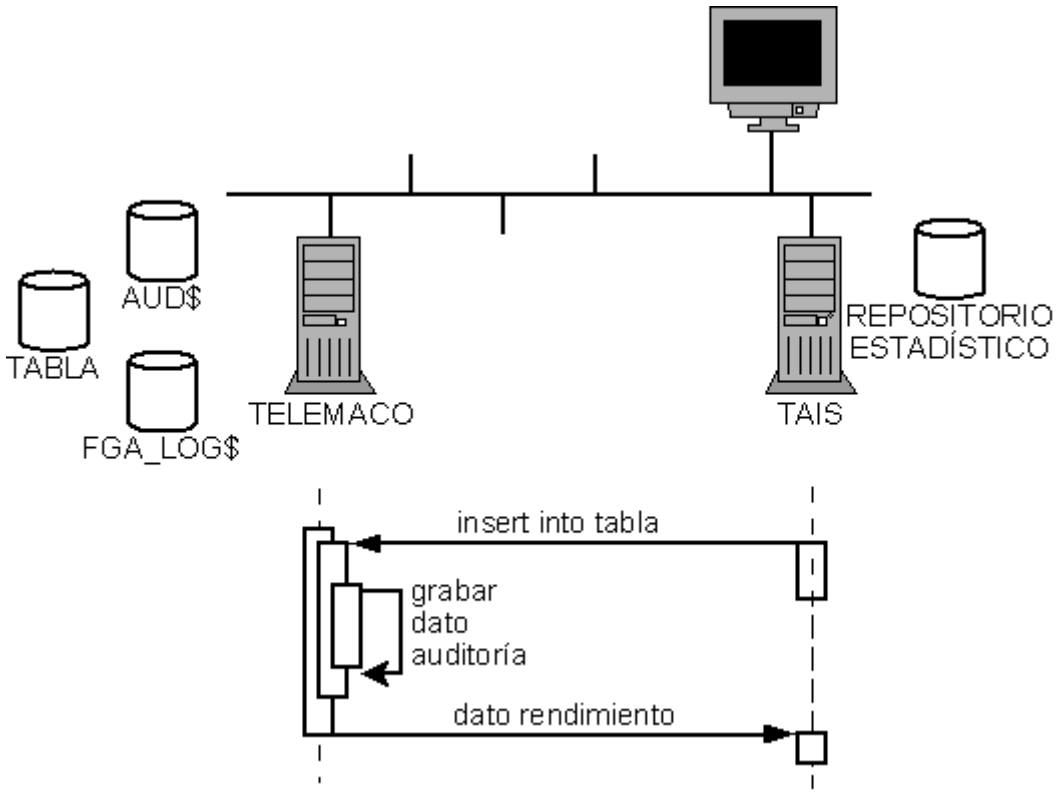

Ilustración 13: Infraestructura para la realización de pruebas de auditoría

El ordenador llamado TELEMACO hará el papel de servidor de la base de datos. Por ello, tendrá un sistema operativo servidor. Se ha elegido Windows Server 2003 *Enterprise Edition Service Pack 1* debido a que este proyecto va destinado a medir el rendimiento en la plataforma Windows. Debido a problemas de compatibilidad con la herramienta de medición de rendimiento *Performance Analysis*, no se ha podido utilizar Windows Server 2008. TELEMACO tendrá instalada la versión 11.1.0.2 de *Oracle* 11g, en la que se encuentra un usuario *mario* que tendrá una tabla que será la que se vaya a auditar. La CPU constará de un procesador Intel® Pentium® 4 a 3,01 GHz con 1,50 GB de RAM y 111 GB de disco duro.

El ordenador llamado TAIS hará el papel de cliente de la base de datos, con SQL+ instalado para el lanzamiento de las pruebas. También tendrá instalada una

pequeña base de datos, con un usuario llamado *auditor* para la realización de las pruebas de backup (pruebas 5 y 6). La CPU constará de un procesador AMD Athlon™ XP 2600+ a 2,08 GHz con 2,00 GB de RAM y 76.3 GB de disco duro.

En cuanto a la medición de rendimiento, el ordenador llamado TAIS tendrá un proceso intermedio (*Middleware*) que recogerá los datos de rendimiento enviados por TELEMACO y los mostrará en forma de gráfica con el *Performance Analysis*. Por su parte, TELEMACO tendrá instalado un proceso llamado Colector (*Collector*) que se encargará de medir la información de rendimiento y de enviársela a TAIS frecuentemente.

A continuación se describirán las pruebas realizadas con la actual infraestructura. Las pruebas se han dividido en 6 grupos según la operación que se vaya a realizar:

- Prueba 1: operación de inserción. Se han realizado dos pruebas con esta operación para ver la diferencia de rendimiento.
    - Prueba 1-A: se trata de realizar inserciones sin parar desde SQL+ en una sola sesión. Esta prueba simulará una carga de datos.
    - Prueba 1-B: consiste en realizar inserciones sin parar, desde SQL+, ejecutando una confirmación (*commit*) tras cada inserción. Esta prueba simulará la continua inserción de registros en diferentes sesiones.
- Prueba 2: operación de consulta. Consiste en realizar consultas continuas, desde un programa de desarrollo propio escrito en Java. Esta decisión es fruto de que, en SQL+, una consulta con muchos registros puede tardar varios segundos, incluso minutos si el número es muy grande, en la salida por pantalla. En lugar de ello, Java

maneja conjuntos de resultado (*ResultSet*), más apropiado a nuestro objetivo. Cada consulta tendrá un número aleatorio de registros de retorno, con un máximo de 100, para que el proceso de envío de datos no normalice el rendimiento del ordenador.

- Prueba 3: operación de modificación. Se han realizado dos pruebas con esta operación para ver la diferencia de rendimiento.
    - Prueba 3-A: se trata de realizar modificaciones sin parar desde SQL+ en una sola sesión.
    - Prueba 3-B: consiste en realizar modificaciones sin parar, desde SQL+, ejecutando una confirmación (*commit*) tras cada modificación. Esta prueba simulará las continuas modificaciones de registros en diferentes sesiones.
- Prueba 4: operación de borrado. Se han realizado dos pruebas con esta operación para ver la diferencia de rendimiento.
    - Prueba 4-A: se trata de realizar borrados sin parar desde SQL+ en una sola sesión.
    - Prueba 4-B: consiste en realizar borrados sin parar, desde SQL+, ejecutando una confirmación (*commit*) tras cada borrado. Esta prueba simulará los continuos borrados de registros en diferentes sesiones.
- Prueba 5: operación de copia y backup. Esta prueba trata de realizar un backup de la tabla de auditoría, ya sea AUD$ para auditoría genérica o FGA_LOG$ para auditoría de grano fino. Se pondrá un límite inicial de 2.500.000 de registros y habrá eventos que guarden registros de auditoría en la tabla sin parar.

Para tener el menor número de registros perdidos, se ha definido otra tabla, con un único campo con un único registro en el que se almacena el momento límite de los registros que van a ser copiados. Los registros que llegues después de ese momento, deberán esperar al siguiente backup. Luego se copian en una tabla auxiliar los registros que hayan llegado antes del momento guardado y se eliminarán dichos registros de la tabla de auditoría original.

Se han realizado dos pruebas con esta operación para ver la diferencia de rendimiento.

  - Prueba 5-A: consiste en realizar la copia de la tabla de auditoría en otro ordenador, de forma que el otro ordenador sea el encargado de realizar el backup. Se simulará la situación en que el auditor tenga un ordenador para él, haciendo único para el desarrollo de su trabajo.

  - Prueba 5-B: se trata de simular la situación en que el auditor o posee un ordenador para él, sino que tiene que trabajar sobre el servidor. Se realizará un backup de la tabla auxiliar en el mismo servidor.

- Prueba 6: operación de restauración de backup. Esta prueba simula a situación en la que el auditor debe trabajar directamente sobre el servidor sin tener un ordenador propio, y trata de restaurar un backup en el servidor.

- Prueba 7: esta prueba compara la evolución del tablespace que contiene la tabla de auditoría genérica con el volumen del que contiene la tabla de auditoría de grano fino con el objetivo de comparar la utilización de las dos auditorías.

La tabla sobre la que se van a crear las auditorías es una tabla de longitud media, en la que hay registros numéricos, registros pequeños y registros grandes para simular descripciones, como currículums, historiales, etc. Tiene la siguiente definición:

```
create table tabla (
 id int primary key,
 dni int,
 num float,
 nombre varchar2(50),
 apellido1 varchar2(50),
 email varchar2(100),
 email2 varchar2(100),
 direccion varchar2(200),
 direccion2 varchar2(200),
 descripcion varchar2(4000)
);
```

Las gráficas que se muestran de cada prueba son las que ofrece *Performance Analysis* de *Quest*.

5.2. Rendimiento del sistema gestor con auditoría genérica

Intentando ver las diferencias de rendimiento entre realizar las pruebas sin auditoría y realizar las pruebas con auditoría genérica, se ha visto que la diferencia es bastante aceptable.

La sentencia de auditoría ejecutada para realizar las pruebas es la siguiente:

```
 audit insert, select, update, delete on mario.tabla by
access;
```

Se puede ver que las operaciones físicas en disco de entrada y salida crecen, pero en un porcentaje lo suficientemente bajo como para no ser necesarios discos extras, al menos si no hay nada más ejecutándose dentro del servidor. Además, esa subida de operaciones se reparte equitativamente entre Redo y datos.

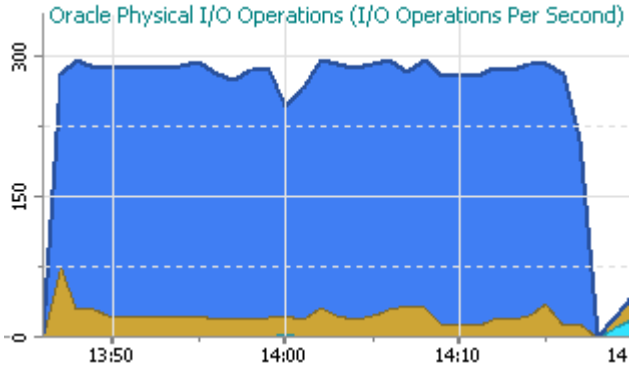

Gráfico I: Operaciones físicas en disco de entrada y salida en la Prueba 1B sin auditoría

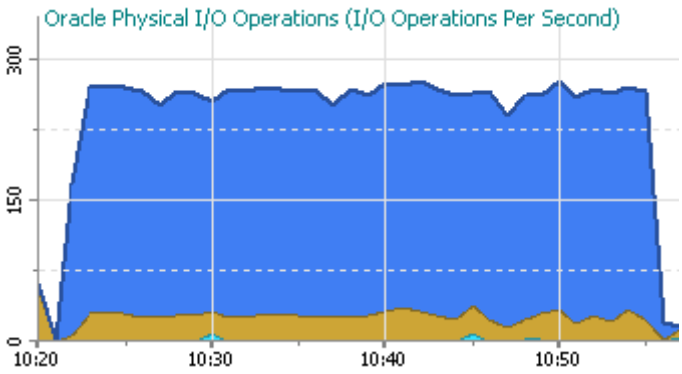

Gráfico II: Operaciones físicas en disco de entrada y salida en la Prueba 1B con auditoría genérica

En todas las pruebas, a todos los niveles, ocurre algo así. Hay más carga en discos, pero poco perceptible. También puede notarse un cierto decremento en el número de paquetes enviados y recibidos vía *Ethernet*, pero no es muy significativo.

Por tanto se puede decir que no hay diferencias claramente significativas entre el uso de la auditoría genérica y el uso de la no auditoría.

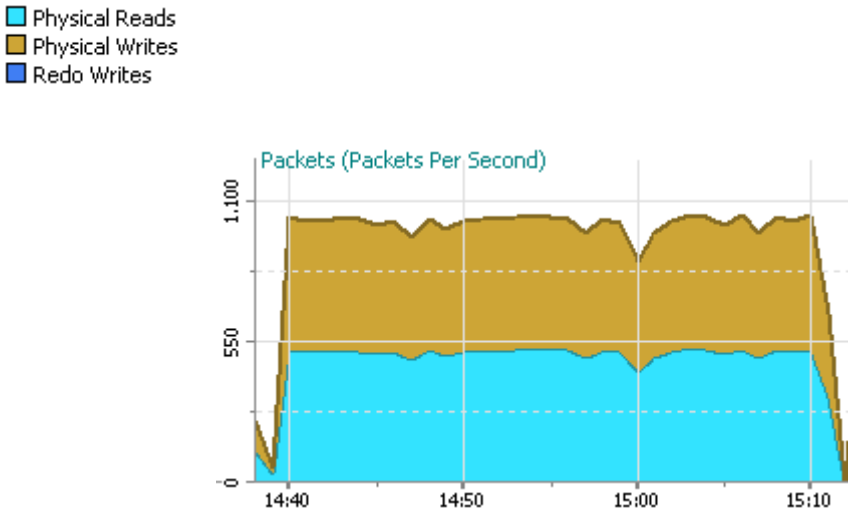

Gráfico III: Paquetes de entrada y salida de red de la Prueba 3B sin auditoría

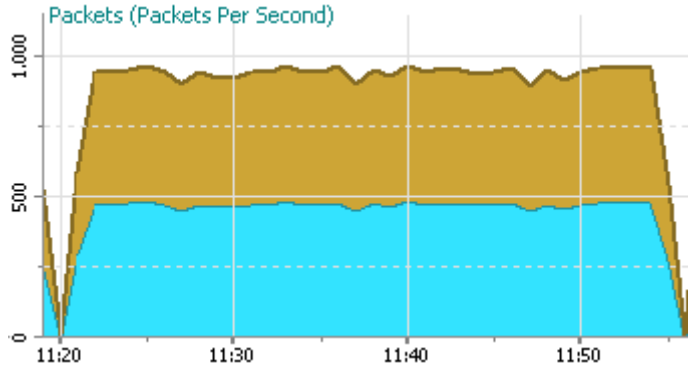

Gráfico IV: Paquetes de entrada y salida de red de la Prueba 3B con auditoría genérica

A continuación se muestra una tabla donde se muestra el tiempo, en minutos, que se ha tardado en hacer cada prueba con auditoría genérica comparándo con el tiempo que se ha tardado en hacer cada prueba sin auditoría.

Con este tiempo se puede ver que la prueba que más aumento de tiempo ha recibido es la prueba 2 (consultas).

PRUEBA	SIN AUDITORÍA	CON AUDITORÍA GENÉRICA	% INCREMENTO DE TIEMPO
1-A: inserción	22	23	4,55%
1-B: inserción + confirmación	34	34	0%
2: consulta	23	26	13,04%
3-A: modificación	23	24	4,35%
3-B: modificación + confirmación	36	37	2,78%
4-A: borrado	22	23	4,55%
4-B: borrado + confirmación	34	35	2,94%

Tabla 1: Tabla de tiempos de las pruebas con auditoría genérica

Este porcentaje que varía de la media de todos los porcentajes es debido a que las consultas no producen escritura en el tablespace UNDO. Al tener que hacer menos escrituras que cualquiera de las otras operaciones, la sentencia se resuelve de forma más rápida. Por ello, al añadir la operación de escritura en la tabla de auditoría, el porcentaje de incremento de tiempo es mayor.

Otro dato que parece interesante es el 0% en la prueba 1-B (inserción + confirmación). Este dato es debido a que la inserción añadiendo la confirmación de operación se realiza en un tiempo muy alto. Por ello, el hecho de añadir una escritura en tabla de auditoría, el porcentaje de incremento de tiempo es muy bajo. No llega a ser 0% completamente, pero es imperceptible.

5.3. Rendimiento del sistema gestor con auditoría de grano fino

Para realizar las pruebas se han tenido que crear cuatro políticas de auditoria de grano fino, cuyo código se describe a continuación:

- Política para auditar la inserción: llamada *politica_insercion*, se crea y se activa con las siguientes sentencias:

```
begin
DBMS_FGA.ADD_POLICY(
object_schema=>'mario',
object_name=>'tabla',
policy_name=>'politica_insercion',
enable=>TRUE,
statement_types=>'insert');
end;
/
```

- Política para auditar la consulta: llamada *politica_consulta*, se crea y se activa con las siguientes sentencias:

```
begin
DBMS_FGA.ADD_POLICY(
object_schema=>' mario',
object_name=>'tabla',
policy_name=>'politica_consulta',
enable=>TRUE,
statement_types=>'SELECT');
end;
/
```

- Política para auditar la modificación: llamada *politica_modificacion*, se crea y se activa con las siguientes sentencias:

```
begin
DBMS_FGA.ADD_POLICY(
object_schema=>' mario',
object_name=>'tabla',
policy_name=>'politica_modificacion',
enable=>TRUE,
statement_types=>'UPDATE');
end;
/
```

- Política para auditar el borrado: llamada *politica_borrado*, se crea y se activa con las siguientes sentencias:

```
begin
DBMS_FGA.ADD_POLICY(
object_schema=> mario',
object_name=>'tabla',
policy_name=>'politica_borrado',
enable=> TRUE,
statement_types=>'DELETE');
end;
/
```

Con estas cuatro sencillas políticas de auditoría podremos realizar las pruebas oportunas, midiendo con idoneidad el rendimiento del servidor cuando se tiene una base de datos Oracle 11g con operaciones auditadas.

La primera diferencia que se ve en auditoría de grano fino es el tiempo de realización de la operación. El siguiente cuadro muestra el tiempo, en minutos, que se ha tardado en hacer cada prueba:

PRUEBA	SIN AUDITORÍA	CON AUDITORÍA DE GRANO FINO	% INCREMENTO DE TIEMPO
1-A: inserción	22	37	68,18%
1-B: inserción + confirmación	40	48	20,00%
2: consulta	23	38	65,22%
3-A: modificación	23	34	47,83%
3-B: modificación + confirmación	36	49	36,11%
4-A: borrado	22	33	50,00%
4-B: borrado + confirmación	34	45	32,35%

Tabla 2: Tabla de tiempos de las pruebas con auditoría de grano fino

Otra de los aspectos a resaltar es el uso de la memoria en disco en la inserción, borrado y consulta. Se puede observar que las escrituras en memoria principal se disparan, pasando del orden de las 25 escrituras al orden de las 225 escrituras, por lo que se puede decir que se incrementa en 9 veces más. Mientras, las lecturas también se incrementan, pero en un orden mucho menor.

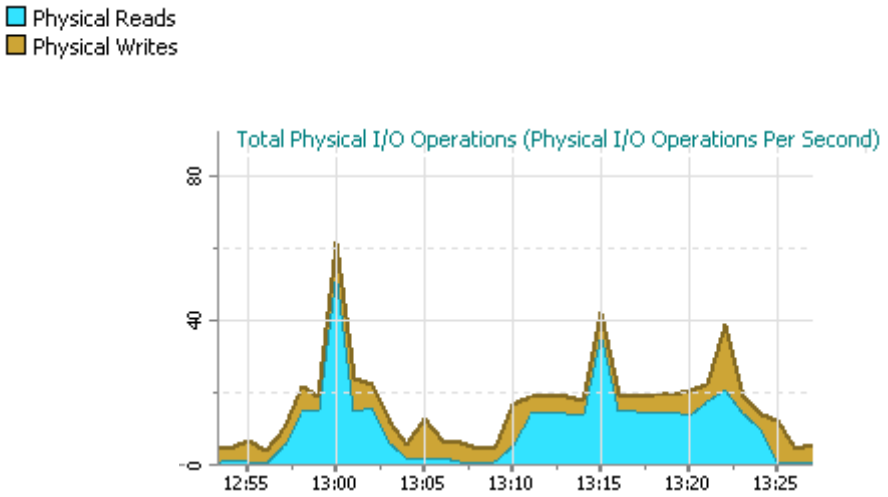

Gráfico V: Operaciones físicas de entrada y salida de la Prueba 2 sin auditoria

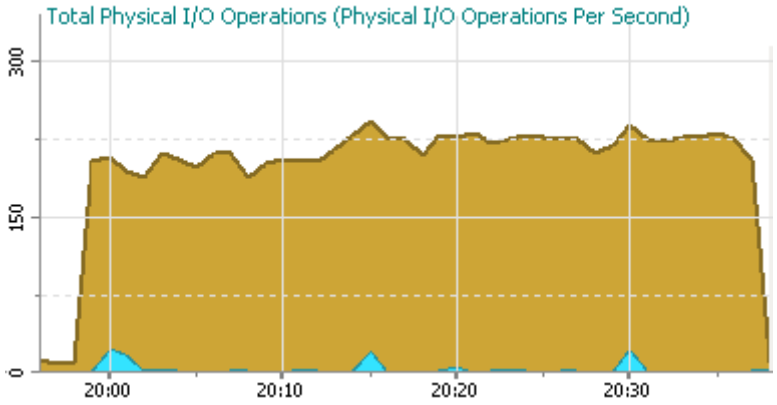

Gráfico VI: Operaciones físicas de entrada y salida de la Prueba 2 con auditoría de grano fino

Además, como se puede ver en la siguiente gráfica, casi un 85% de esas escrituras es sobre los ficheros de Redo, por lo que si el auditor cree conveniente realizar auditoría de grano fino en la inserción, sería muy recomendable que utilizara otro disco para los ficheros de Redo.

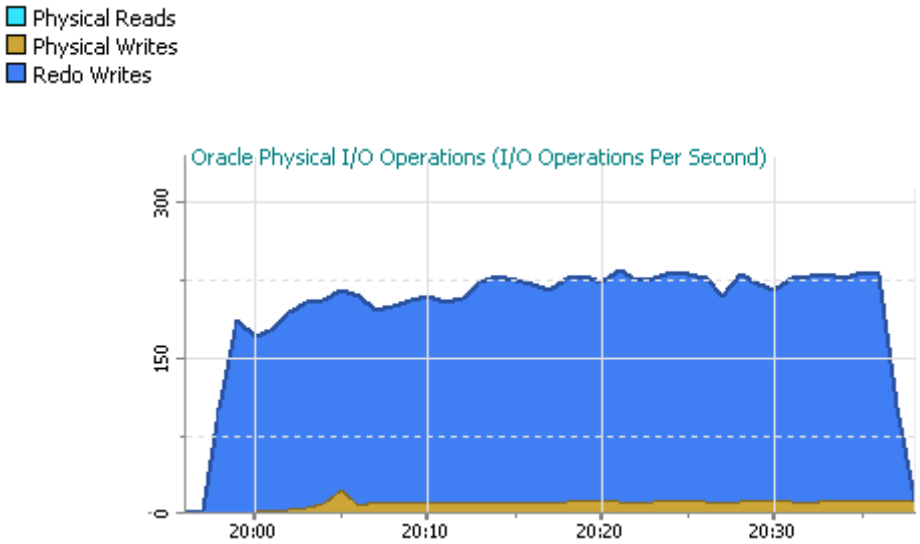

Gráfico VII: Operaciones físicas de Oracle de entrada y salida de la Prueba 2 con auditoría de grano fino

Otro motivo para separar los ficheros de Redo a otro disco está en las consultas, donde ocurre algo similar, en igual proporción.

La diferencia no es tan fuerte en la prueba 1-B, pero el uso reiterado de los ficheros de Redo disparan también la escritura si no hubiera auditoría.

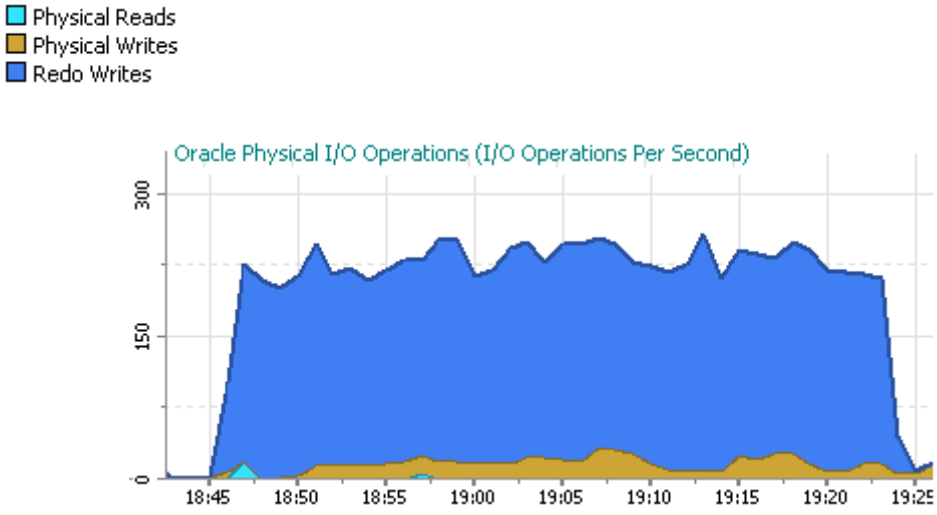

Gráfico VIII: Operaciones físicas de Oracle de entrada y salida de la Prueba 1-B sin auditoría

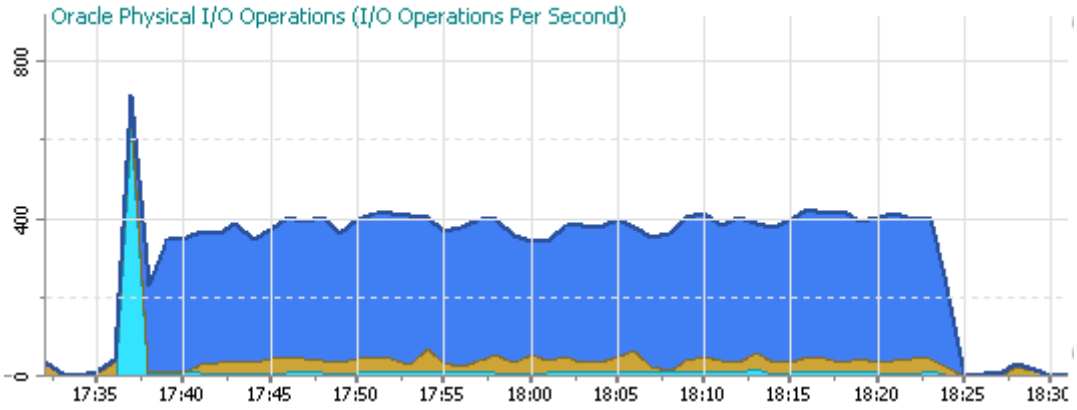

Gráfico IX: físicas de Oracle de entrada y salida de la Prueba 1-B con auditoría de grano fino

Como se puede ver en las siguientes gráficas, el número de operaciones realizadas disminuye debido a que el tiempo que tarda de más en realizar la operación es tiempo de espera en el lanzamiento de inserciones contra la base de datos. Por lo que esta diferencia tiene mucho que ver con el primer cuadro de tiempo de más. Exactamente lo mismo ocurre con el resto de operaciones, lo que es algo lógico.

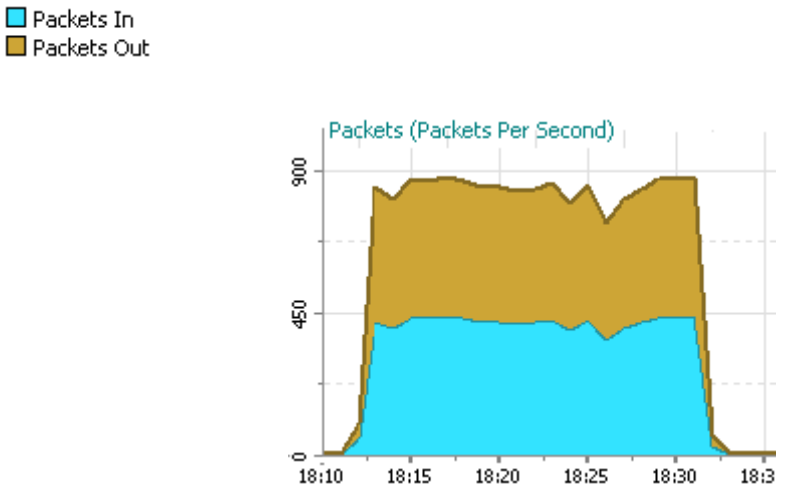

Gráfico X: Paquetes de red enviados y recibidos en la Prueba 1-A sin auditoria

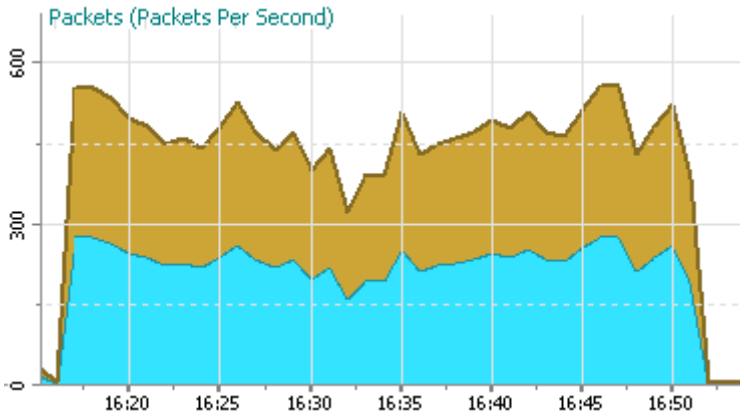

Gráfico XI: Paquetes de red enviados y recibidos en la Prueba 1-A con auditoría de grano fino

En las pruebas 3-A y 3-B se ven picos de lectura en disco. En la prueba 3-A se ve un pico de lectura que se distingue perfectamente con respecto a la gráfica de la misma prueba sin auditoría.

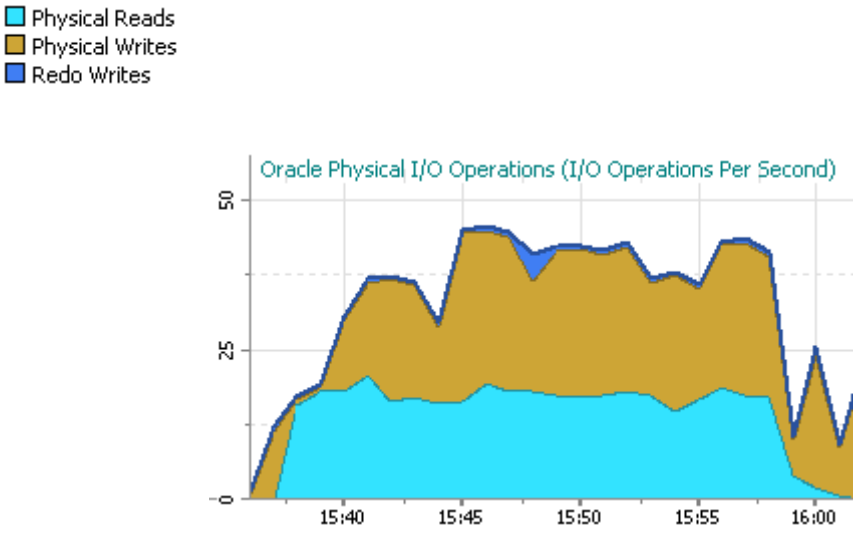

Gráfico XII: Operaciones físicas de Oracle de entrada y salida de la Prueba 3-A sin auditoria

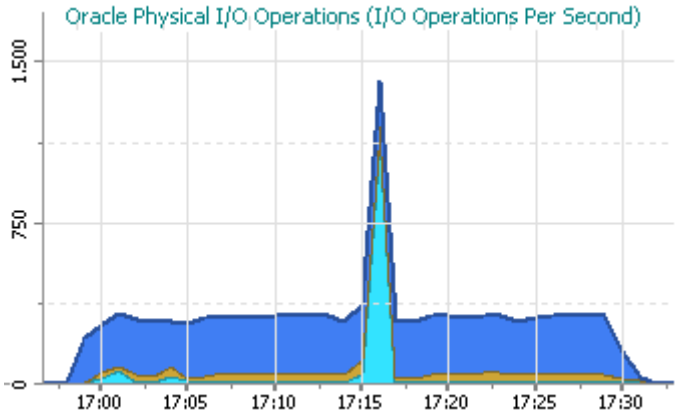

Gráfico XIII: Operaciones físicas de Oracle de entrada y salida de la Prueba 3-A con auditoría de grano fino

En la gráfica sin auditoría de la prueba 3-B se ve también un pico de lectura, lo que es lógico debido al uso masivo de la confirmación de operación *commit*. Sin embargo, en la gráfica con auditoría de grano fino de la prueba 3-B, ese mismo pico se dispara aún más.

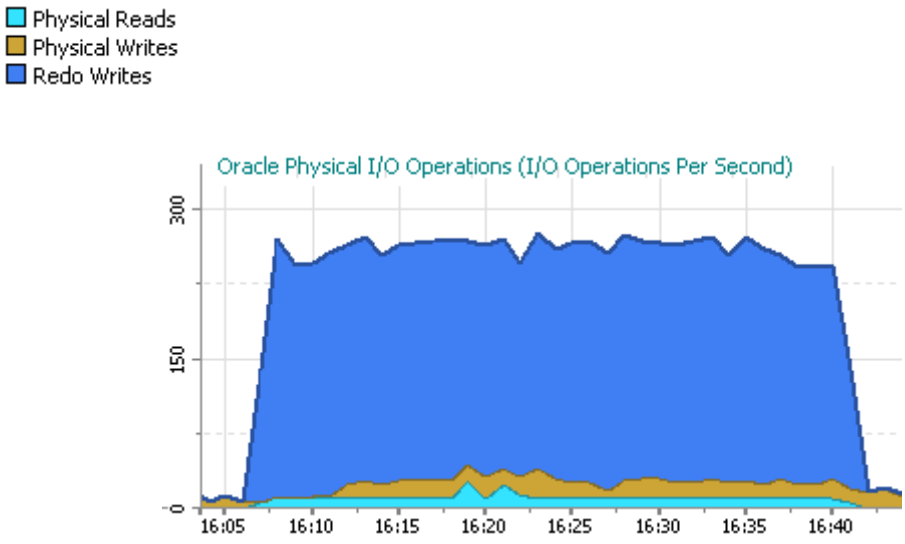

Gráfico XIV: Operaciones físicas de Oracle de entrada y salida de la Prueba 3B sin auditoría

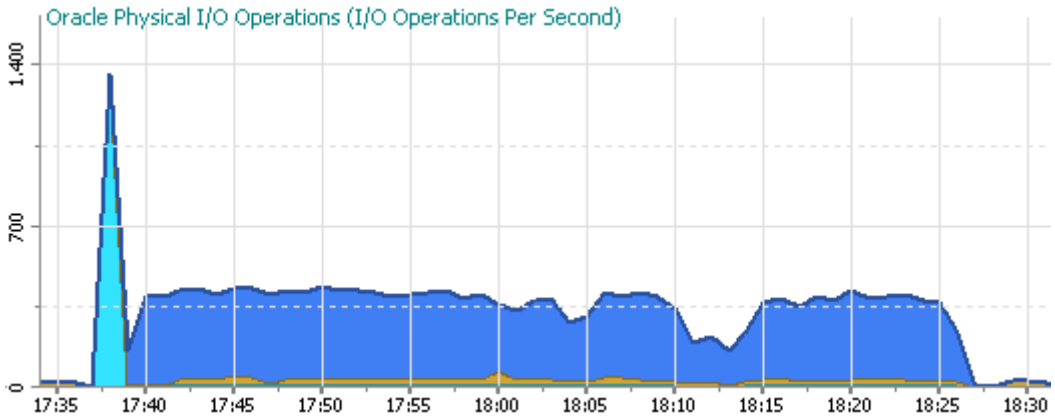

Gráfico XV: Operaciones físicas de Oracle de entrada y salida de la Prueba 3-B con auditoría de grano fino

5.4. Rendimiento del sistema gestor con auditoría genérica y de grano fino

Como en anteriores ocasiones, la primera diferencia es el tiempo de realización de la operación. El siguiente cuadro muestra el tiempo, en minutos, que se ha tardado en hacer cada prueba:

PRUEBA	SIN AUDITORÍA	CON AUDITORÍA GENÉRICA Y DE GANO FINO	% INCREMENTO DE TIEMPO
1-A: inserción	22	36	63,64%
1-B: inserción + confirmación	40	50	25,00%
2: consulta	23	43	86,96%
3-A: modificación	23	36	56,52%
3-B: modificación + confirmación	36	60	66,67%
4-A: borrado	22	37	68,18%
4-B: borrado + confirmación	34	48	41,18%

Tabla 3: Tabla de tiempos de las pruebas con auditoría genérica y de grano fino

Con este rendimiento, el auditor debe ponderar los valores obtenidos con el tipo de uso que se le da a la base de datos.

Otro de los aspectos en cuanto al rendimiento que llama la atención uniendo las auditorías genérica y de grano fino es el uso del procesador. Sobretodo por el hecho de que, por separado, el uso del procesador no había supuesto nada que resalte respecto al uso del procesador sin auditoría.

En los grupos de pruebas 1, 3 y 4 se ve que el uso del procesador ronda el 50%, rondando el 30% en el uso de Oracle sin auditoría. Esto quiere decir que

usar los dos tipos de auditoría en Oracle puede suponer que otros procesos que marchen en el servidor tengan menos recursos computacionales.

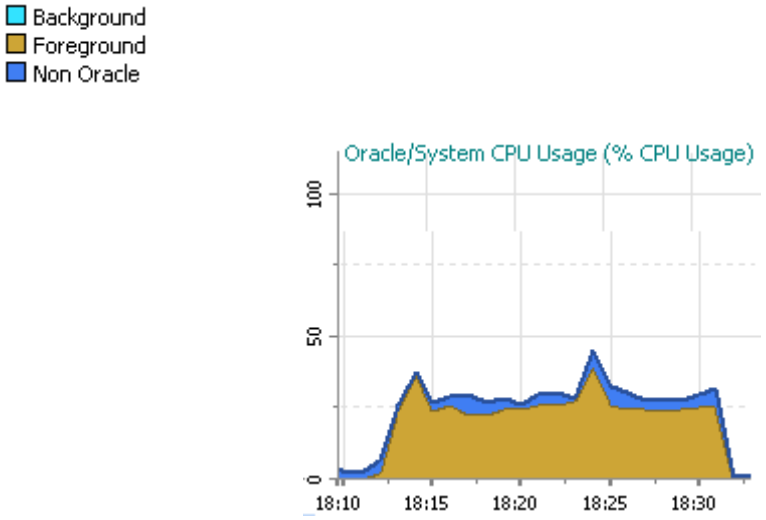

Gráfico XVI: Uso del procesador en la Prueba 1-A sin auditoria

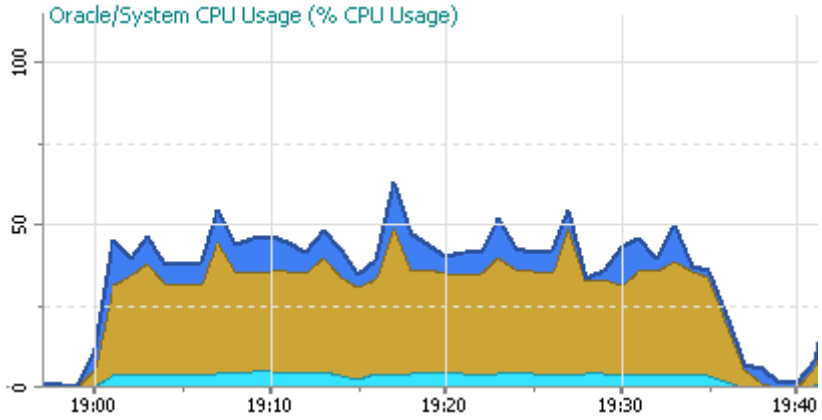

Gráfico XVII: Uso del procesador en la Prueba 1-A con auditoría genérica y auditoría de grano fino

Incluso se ve, sobretodo en la prueba 3-B, que la cola de procesos crece lo suficiente como para que llame la atención.

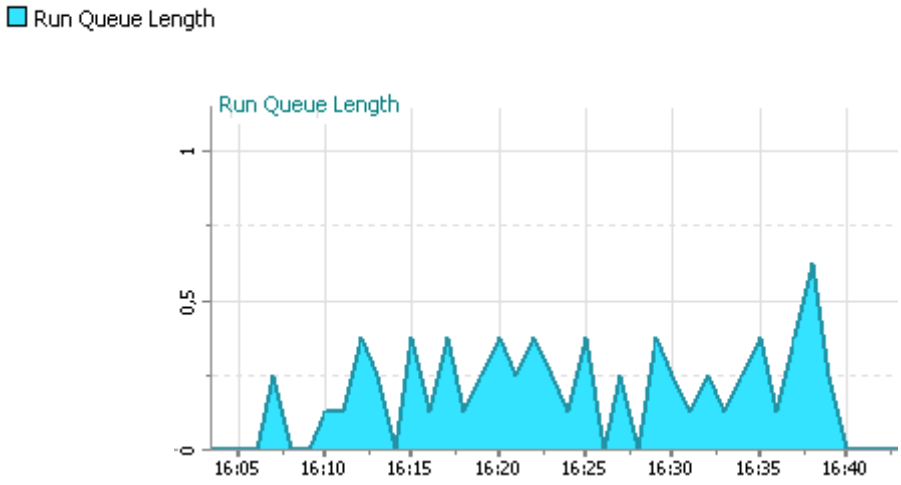

Gráfico XVIII: Uso de la cola de procesos en la Prueba 3-B sin auditoria

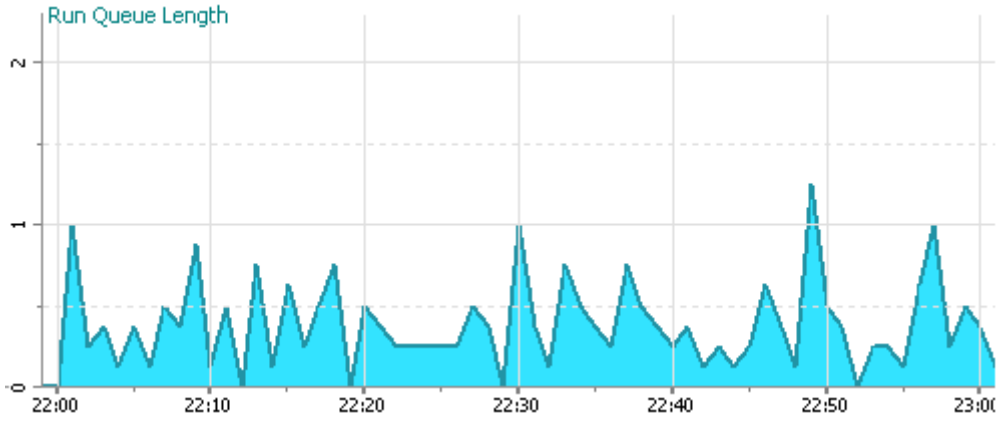

Gráfico XIX: Uso de la cola de procesos en la Prueba 3-B con auditoría genérica y de grano fino

Sin embargo, a menos que el servidor esté compartido por otra aplicación u otras aplicaciones que necesiten el procesador, no se considera lo suficientemente alto como para implantar otro procesador.

Al igual que en ocasiones anteriores, las operaciones físicas de entrada y salida se disparan hasta, incluso, rondar el 60% más de operaciones físicas que con la auditoría de grano fino. No ocurre con las consultas que, aunque crece, no lo hace tanto como con las inserciones, modificaciones y borrados.

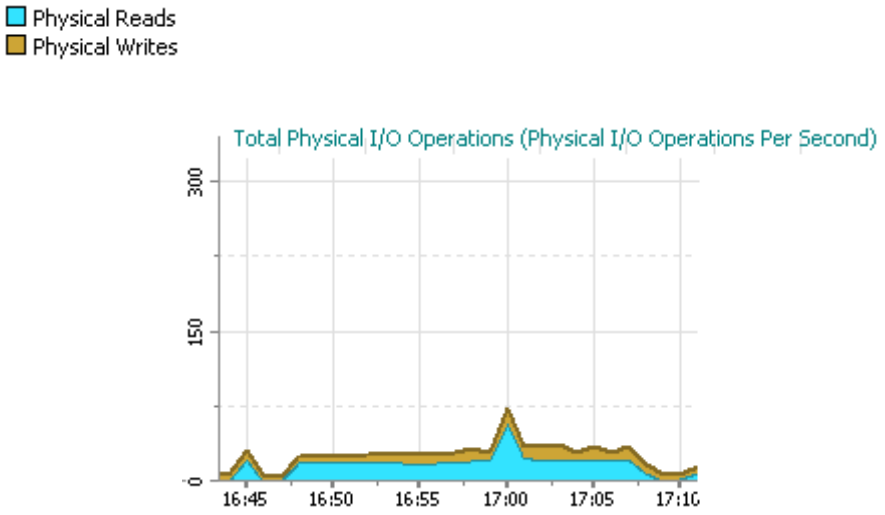

Gráfico XX: Operaciones físicas de entrada y salida de la Prueba 4 sin auditoría

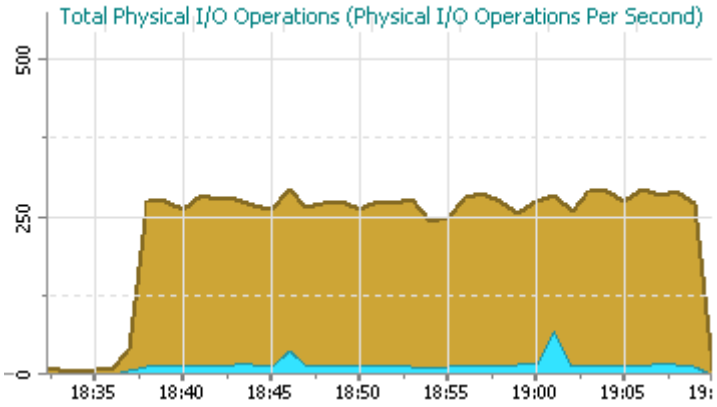

Gráfico XXI: Operaciones físicas de entrada y salida de la Prueba 4 con auditoría de grano fino

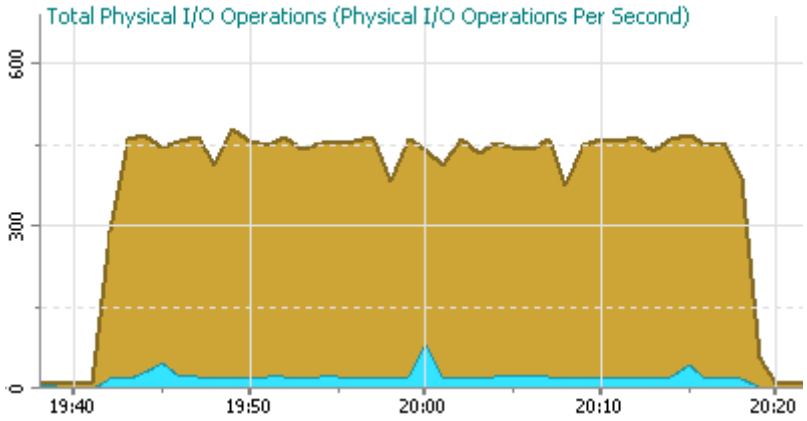

Gráfico XXII: Operaciones físicas de entrada y salida de la Prueba 4 con auditoría genérica y de grano fino

En este caso sí sería muy recomendable reubicar los ficheros de Redo en otro disco, pues la mayoría de las operaciones son en estos ficheros.

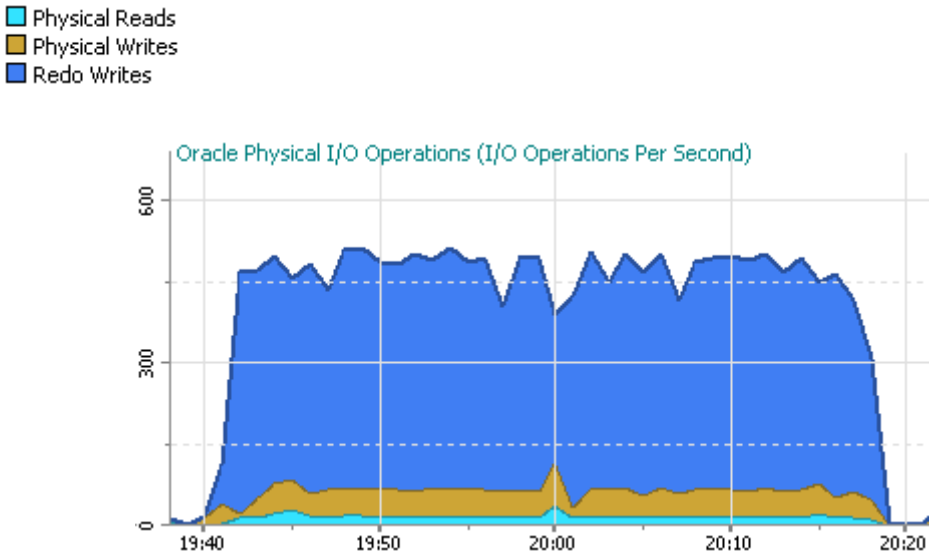

Gráfico XXIII: Operaciones físicas de Oracle de entrada y salida de la Prueba 4 con auditoría genérica y de grano fino

Otro de los aspectos a destacar es el uso de disco en cuanto al porcentaje de tiempo que el ordenador está esperando a que el disco le sirva en operaciones de entrada y salida, sobretodo en las pruebas 2 y 3-B. Como se ve en las siguientes gráficas, hay momentos en los que aparece un cuello de botella que no aparece si usáramos sólo auditoría genérica o auditoría de grano fino.

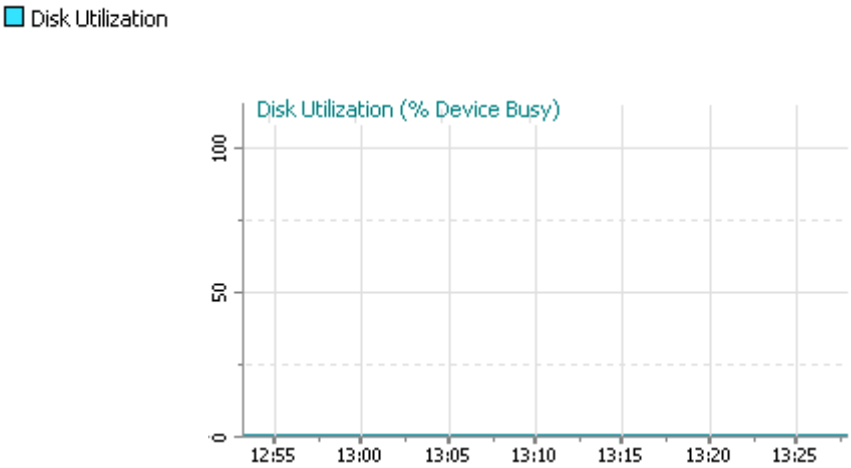

Gráfico XXIV: Porcentaje de tiempo de espera del procesador a la respuesta del disco en la Prueba 2 sin auditoría

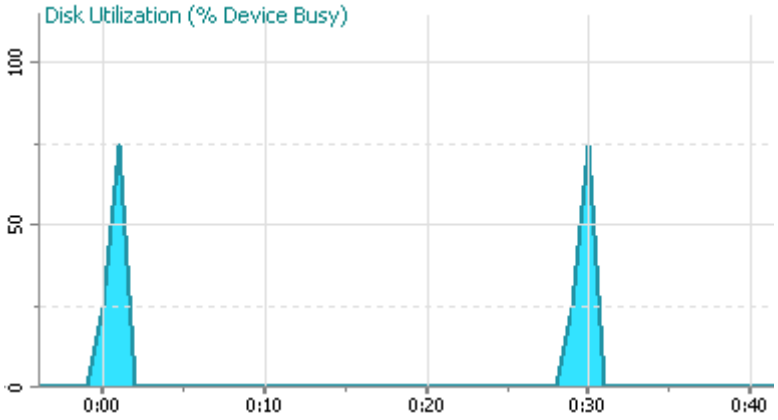

Gráfico XXV: Porcentaje de tiempo de espera del procesador a la respuesta del disco en la Prueba 2 con auditoría genérica y de grano fino

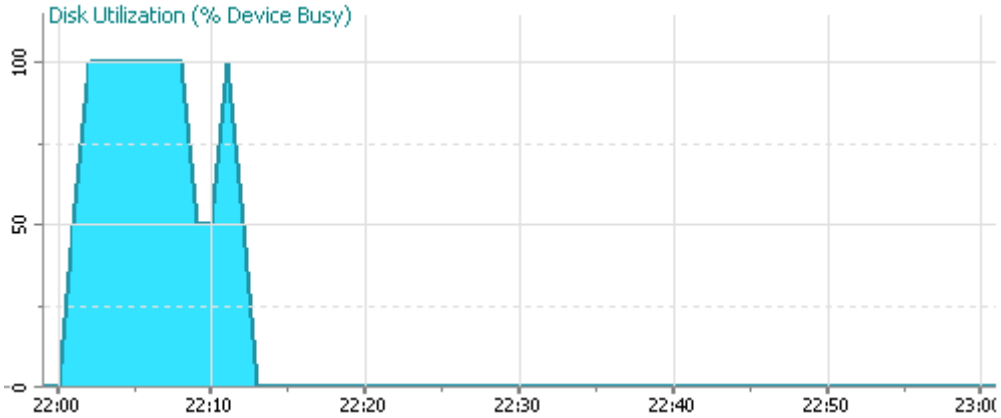

Gráfico XXVI: Porcentaje de tiempo de espera del procesador a la respuesta del disco en la Prueba 3-B con auditoría genérica y de grano fino

Por otro lado, debido a que una operación tarda más tiempo en ser ejecutada que si no hubiese auditoría, ocurre lo mismo en el envío y recepción de paquetes de red que en ocasiones anteriores, sólo que esta vez la diferencia es más amplia, pues las operaciones se hacen esperar más, con un porcentaje acorde al tiempo de ejecución de la operación. La siguiente gráfica muestra el envío y recepción de paquetes de red en la prueba 1-A con la auditoría genérica y con la auditoría de grano fino. Para comparar puede ver el lector la gráfica VI

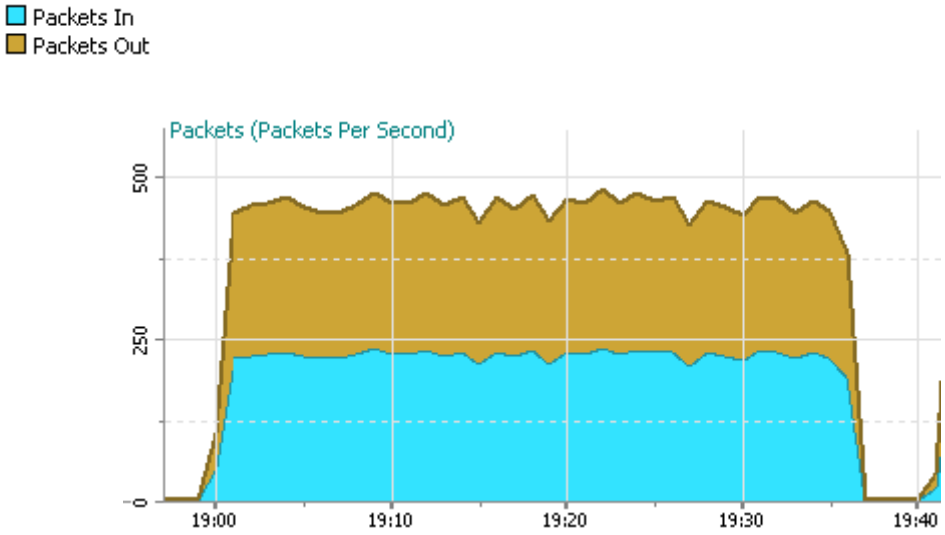

Gráfico XXVII: Paquetes de red enviados y recibidos en la Prueba 1-A con auditoría genérica y de grano fino

Para terminar, sólo destacar el uso de los ficheros de Redo en los grupos de prueba 1, 3 y 4. Supone un incremento de alrededor del 50% en las operaciones con confirmación *commit* respecto a la auditoría de grano fino, que a su vez supone un incremento del 30% sobre la misma prueba realizada sin auditoría…

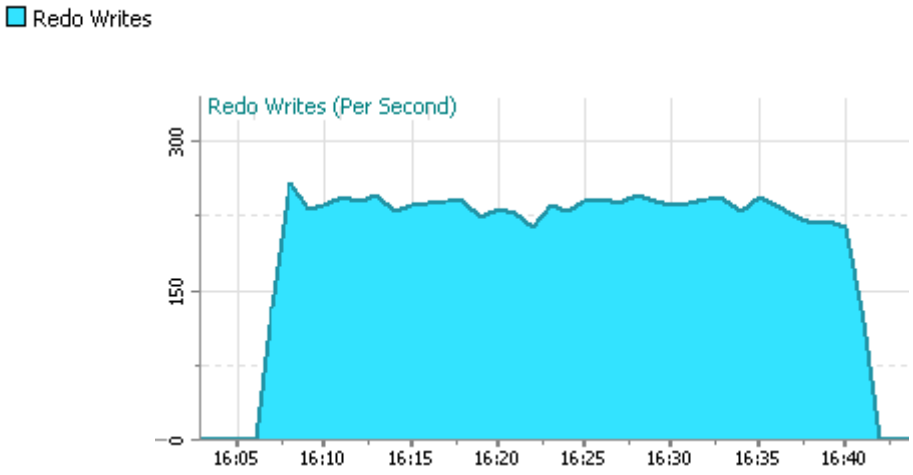

Gráfico XXVIII: Escrituras en ficheros de Redo en la Prueba 3-B sin auditoría

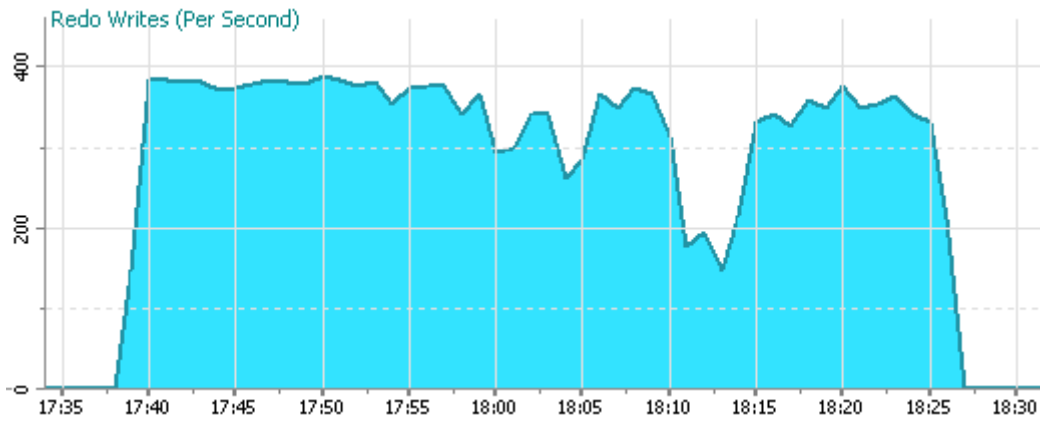

Gráfico XXIX: Escrituras en ficheros de Redo en la Prueba 3-B con auditoría de grano fino

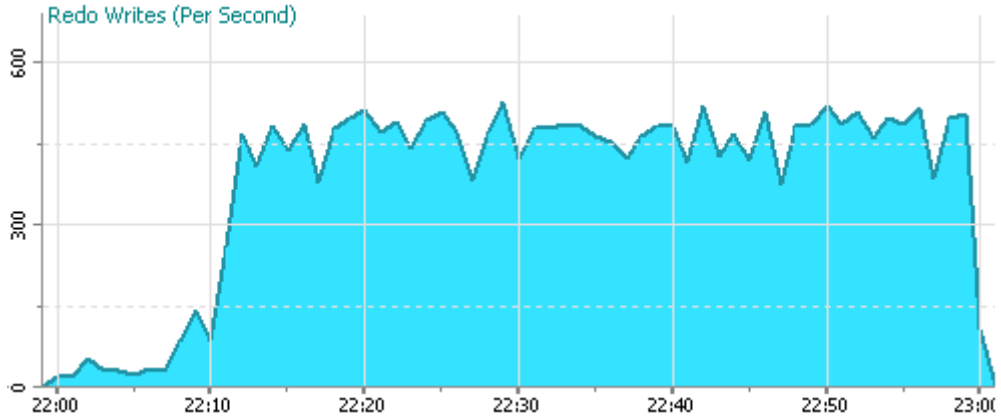

Gráfico XXX: Escrituras en ficheros de Redo en la Prueba 3-B con auditoría genérica y de grano fino

…y un incremento del 75% respecto a la auditoría de grano fino, que a su vez supone un incremento del 200 por uno sobre la misma prueba realizada sin auditoría.

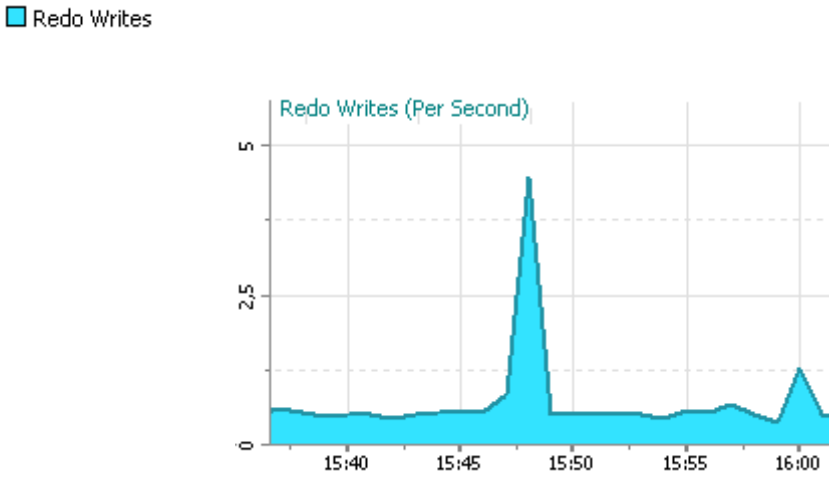

Gráfico XXXI: Escrituras en ficheros de Redo en la Prueba 3-A sin auditoría

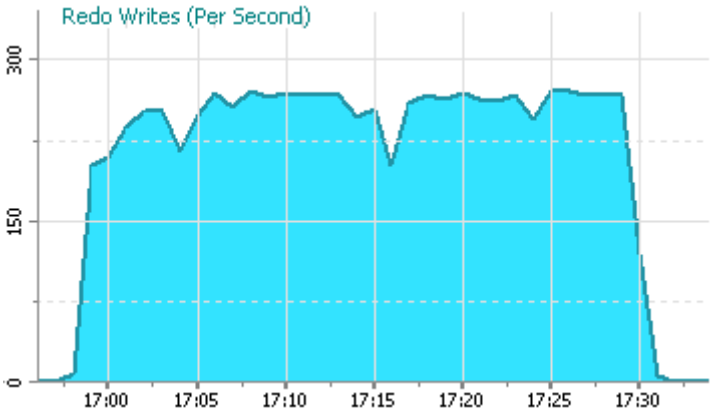

Gráfico XXXII: Escrituras en ficheros de Redo en la Prueba 3-A con auditoría de grano fino

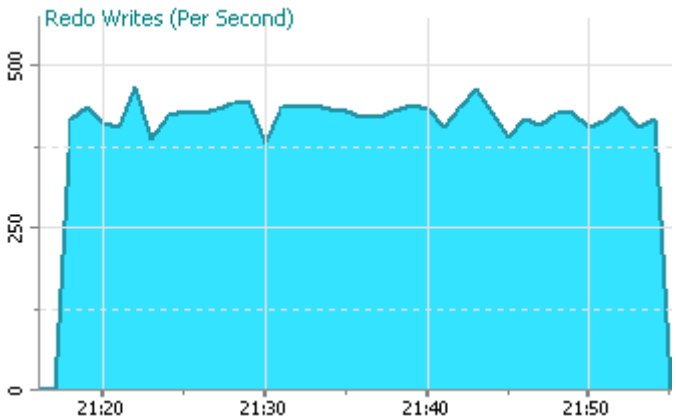

Gráfico XXXIII: Escrituras en ficheros de Redo en la Prueba 3-A con auditoría genérica y de grano fino

Todo esto aumenta el grado de recomendación de separar los ficheros de Redo en discos distintos.

5.5. Rendimiento de espacio de tablas

En este punto se ha realizado la prueba 7. Esta prueba consiste en crear una auditoría genérica y una auditoría de grano fino de tal forma que se audite lo mismo. Como la auditoría de grano fino consume espacio de tabla dependiendo de la sentencia que se audite (cuanto más larga sea la sentencia, más espacio consumirá) debido a que la tabla FGA_LOG$ tiene un campo llamado SQLTEXT que contiene toda la sentencia auditada, se ha decidido auditar la sentencia más corta que, a nivel de empresa, se puede realizar:

```
select * from tabla where id > xx and id < yy;
```

donde *xx* e *yy* son números aleatorios entre 1 y 10.

El resultado ha sido la siguiente gráfica, donde la línea roja indica el volumen del espacio de tablas que contiene la tabla de auditoría genérica y la línea azul indica el volumen del espacio de tablas que contiene la tabla de auditoría de grano fino. La variable independiente está representada por el número de eventos realizados para que se guarden registros de auditoría. La variable dependiente está representada por el volumen disponible del espacio de tabla, en bytes.

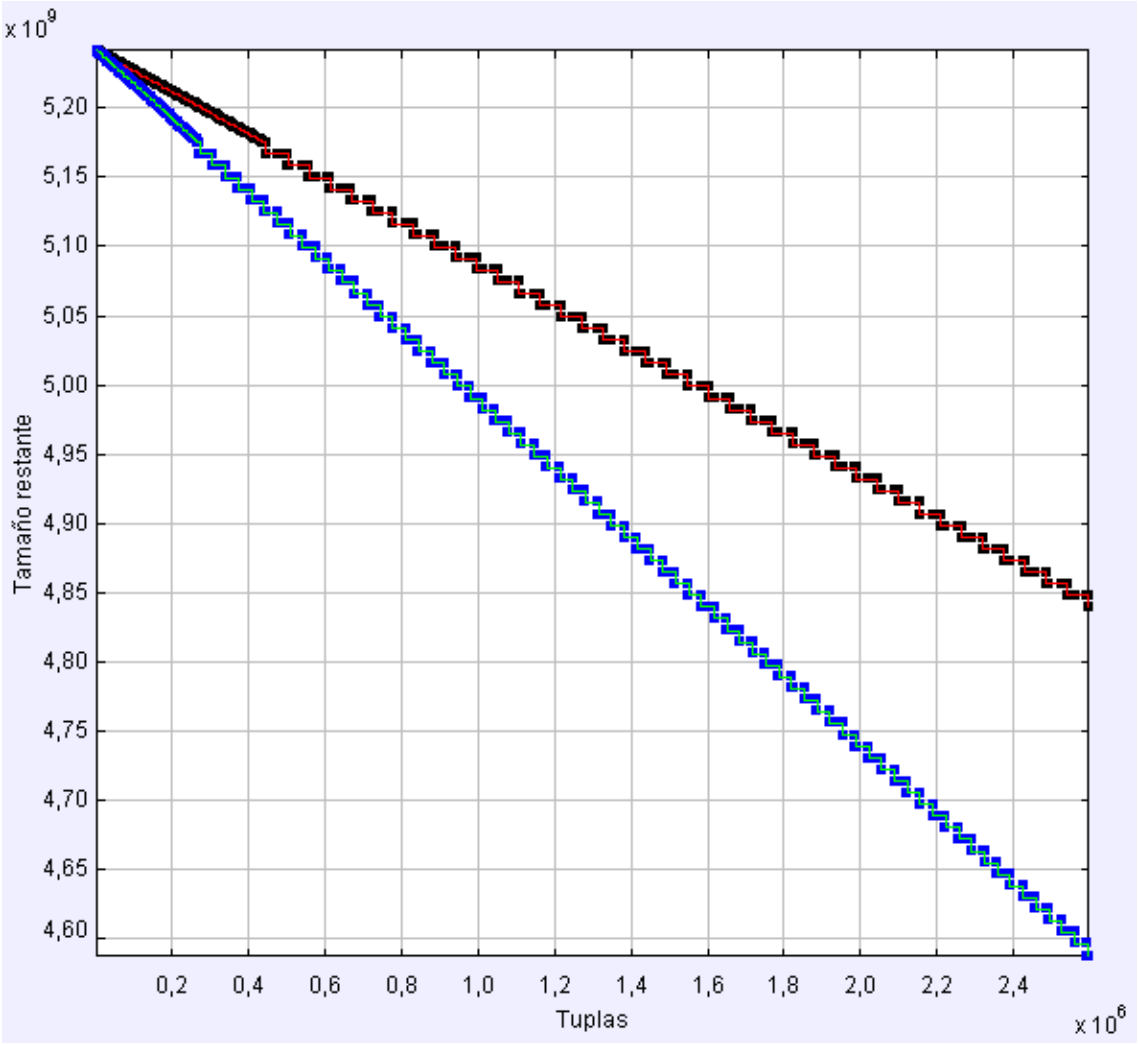

Gráfico XXXIV: Evolución del espacio disponible en los espacios de tablas de auditoría

Se puede ver que, desde un inicio, el volumen libre en el espacio de tabla de auditoría de grano fino decrece de forma mucho más rápida que el volumen libre del espacio de tabla de auditoría genérica. Ambos decrecen de forma lineal, pero la pendiente de la línea azul es más pronunciada.

En el siguiente gráfico se muestra la relación que hay entre las dos evoluciones en referente a la disminución del tamaño. Se puede ver que dicha disminución del tamaño tiende a que el espacio de tablas que contiene la tabla de auditoría de grano fino disminuye a una velocidad de 1,65 veces la velocidad de disminución del espacio de tablas que contiene la tabla de auditoría genérica:

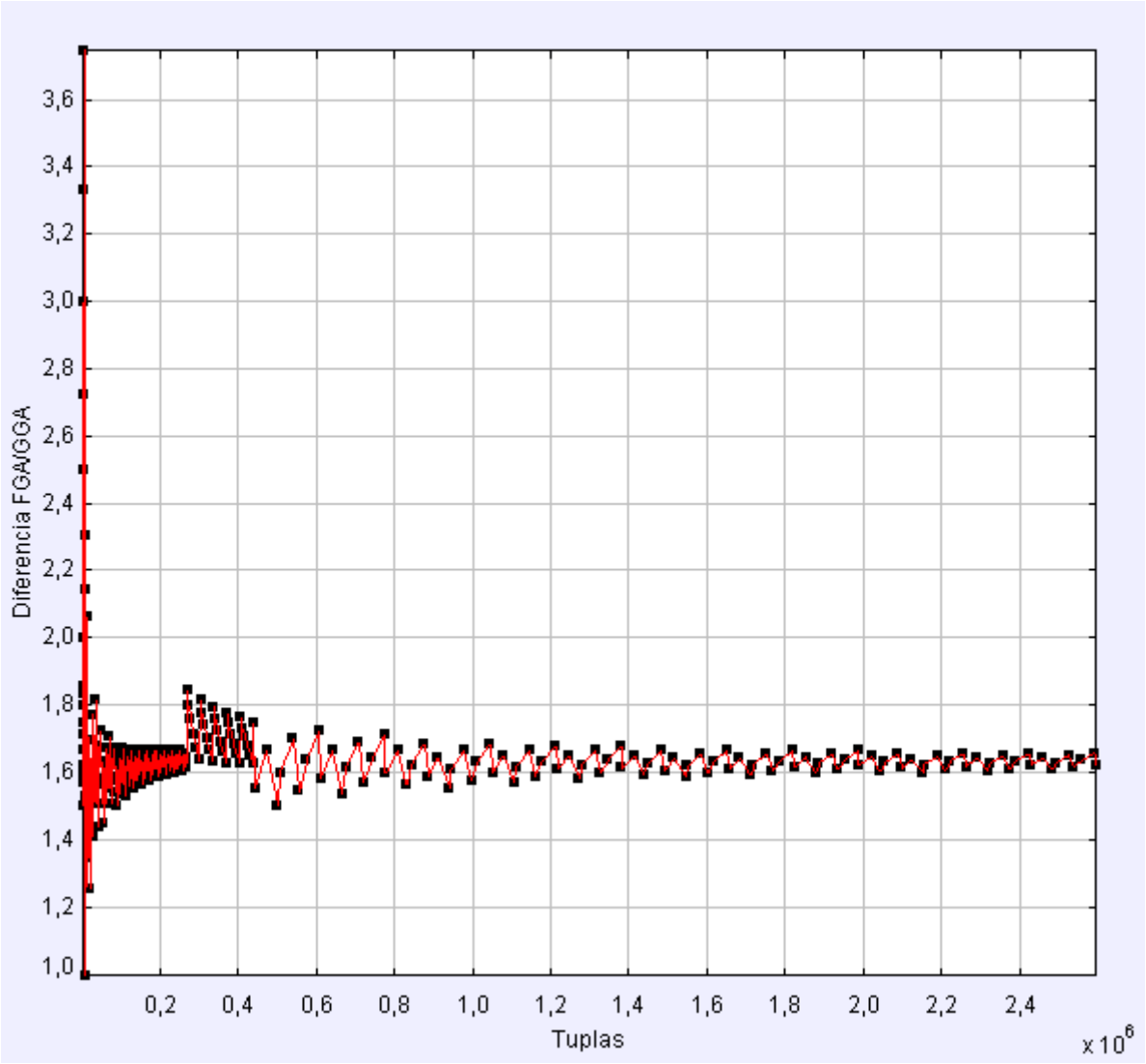

Gráfico XXXV: Relación de crecimiento de información en el espacio de tablas que contiene la tabla de auditoría de grano fino con relación al crecimiento de información en el espacio de tablas que contiene la tabla de auditoría genérica

5.6. Rendimiento de copia, backup y recuperación

En este punto se estudiará lo obtenido en las pruebas 5-A (copia y backup en el mismo servidor), 5-B (copia en PC Auditor) y 6 (recuperación de backup en el mismo servidor). Estas pruebas determinarán la infraestructura para realizar la auditoría desde el punto de vista del auditor: una infraestructura no sólo para auditar sentencias, sino para realizar backup y recuperación de las sentencias auditadas.

Lo primero que llama la atención es el hecho de que la prueba 5-B no haya ido bien para la auditoría de grano fino. Mientras se realizaba la copia de la tabla en PC Auditor, se seguían insertando registros de auditoría de grano fino en FGA_LOG$. Este hecho desembocó en un error en la base de datos: se cortó la comunicación con el cliente y se paró la copia. Esto es debido a la alta cantidad de información que maneja la tabla FGA_LOG$: la velocidad de entrada de registros de auditoría de grano fino era mayor que la velocidad con que se enviaban registros a la otra base de datos.

Esto hace que la base de datos se colapse. Las siguientes gráficas muestra la velocidad real de entrada de operaciones y la velocidad de entrada de operaciones en la Prueba 5-B.

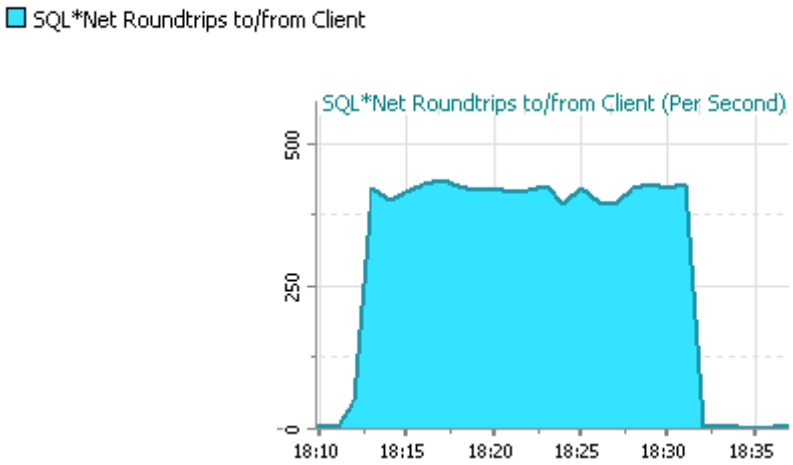

Gráfico XXXVI: Velocidad de entrada de operaciones a la base de datos sin auditoría

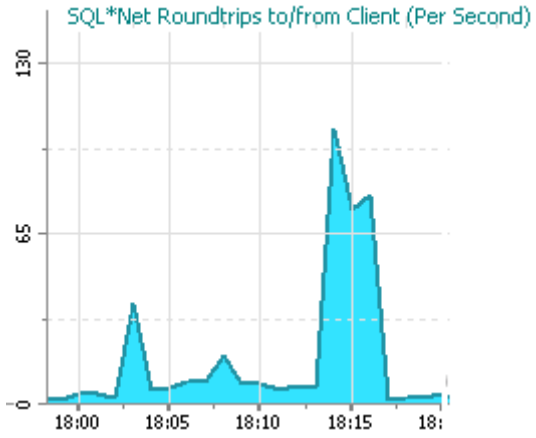

Gráfico XXXVII: Velocidad de entrada de operaciones a la base de datos con auditoría de grano fino mientras se copia la tabla de auditoría en otro ordenador

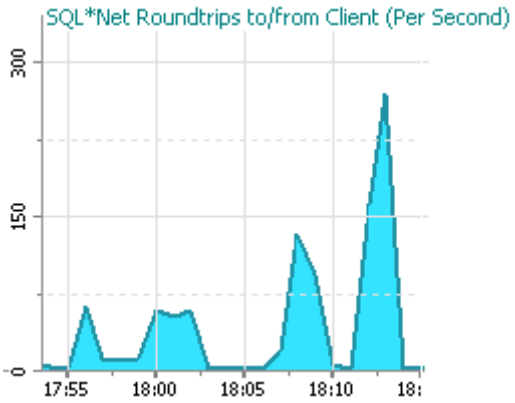

Gráfico XXXVIII: Velocidad de entrada de operaciones a la base de datos con auditoría de grano fino mientras se copia la tabla de auditoría en el mismo ordenador

Por tanto, para hacer copia y backup con auditoría de grano fino desde otro ordenador, hay que hacerlo cada muy poco tiempo (tener menos registros de auditoría), lo que lo hace menos eficiente, o tener un ancho de banda mayor que el ancho de banda de una red Ethernet, lo que lo hace muy difícil.

En conclusión, es más recomendable realizar copia y backup en el mismo ordenador si se realiza auditoría de grano fino.

La prueba 5-A presenta mejores resultados para la auditoría genérica que la prueba 5-B. Por un lado, hay que destacar el hecho de que tarda la mitad de tiempo: 7 minutos para la prueba 5-A y 15 minutos para la prueba 5-B.

Además se puede apreciar un menor uso de los recursos computacionales (uso del procesador).

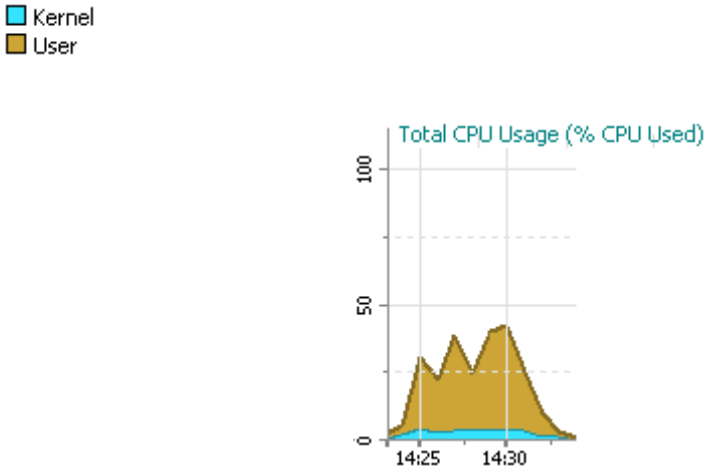

Gráfico XXXIX: Uso de la CPU en la Prueba 5-A para auditoría genérica

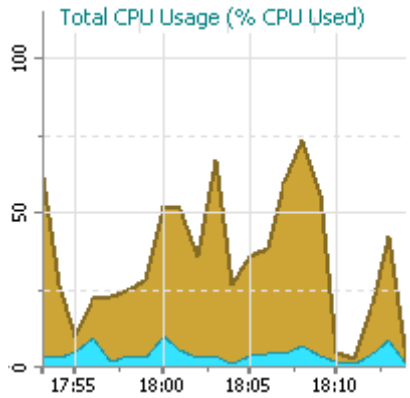

Gráfico XL: Uso de CPU en la Prueba 5-B con auditoría genérica

Además, la prueba 5-A bloquea menos la base de datos con respecto a la prueba 5-B en referencia al número de operaciones que permite realizar, lo que es algo muy importante desde el punto de vista de la disponibilidad de la base de datos. Sin embargo, esto hace que se pierdan muchos más datos de auditoría. Usando la fecha actual de *Oracle* (*sysdate*) como referencia para realizar el backup, se llega al siguiente resultado:

Prueba 5-A:

```
SQL> insert into AUX_AUD$ (SELECT * FROM sys.AUD$@enlace_auditoría_aud WHERE ntimestamp#<=(select hoy from hoy));

2500672 filas creadas.

SQL> DELETE FROM sys.AUD$@enlace_auditoría_aud where ntimestamp# <= (select hoy from hoy);

2554847 filas suprimidas.
```

Esto es una pérdida de más de 50.000 datos de auditoría.

Prueba 5-B:

```
SQL> insert into AUX_AUD$ (SELECT * FROM sys.AUD$ WHERE ntimestamp#<=(select hoy from hoy));

2503324 filas creadas.

SQL> DELETE FROM sys.AUD$ where ntimestamp# <= (select hoy from hoy);

2505250 filas suprimidas.
```

Esto es una pérdida de poco más de 2.000 registros de auditoría.

Recuerde que la pérdida es debido a que la prueba se realiza mediante un *scritp* que va ejecutando operaciones que hacen saltar las políticas de auditoría continuamente.

En conclusión, para bases de datos muy concurrentes, con muchos usuarios y muy complejas, es mejor utilizar el primer método: copia y backup en el mismo servidor. Para bases de datos poco concurrentes, es recomendable utilizar el segundo método: copia y backup en PC Auditor.

En cuanto a la prueba 6, tiene un uso de disco muy grande, por lo que es recomendable, pero no imprescindible, realizar este tipo de operaciones en días o momentos en los que la base de datos tenga menor uso.

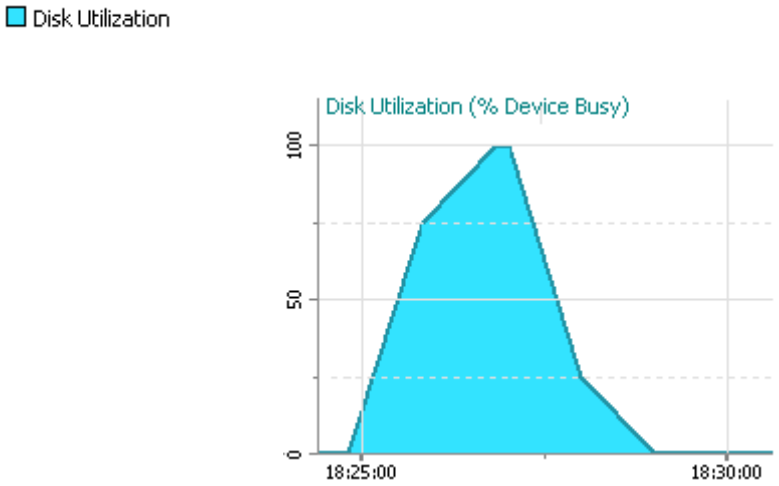

Gráfico XLI: Uso de disco en la Prueba 6 con auditoría genérica

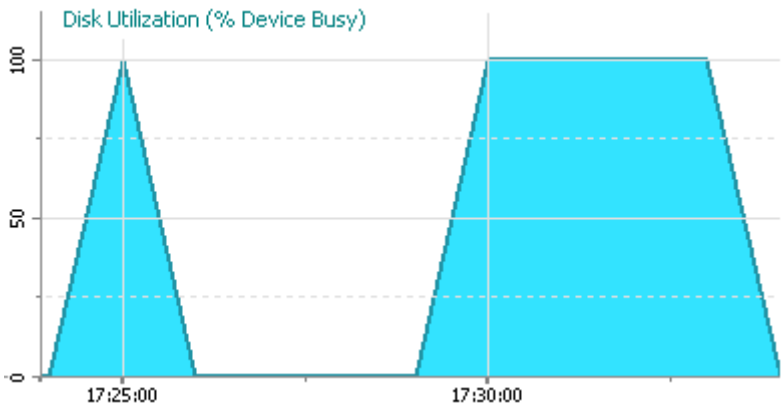

Gráfico XLII: Uso de disco en la Prueba 6 con auditoría de grano fino

Estas acciones son recomendables debido a que este hecho hace que se paren las peticiones desde cliente. El disco tiene tanto uso cuando se carga la tabla que las peticiones se ralentizan. Se puede comprobar comparando el nivel con el nivel de la prueba 2.

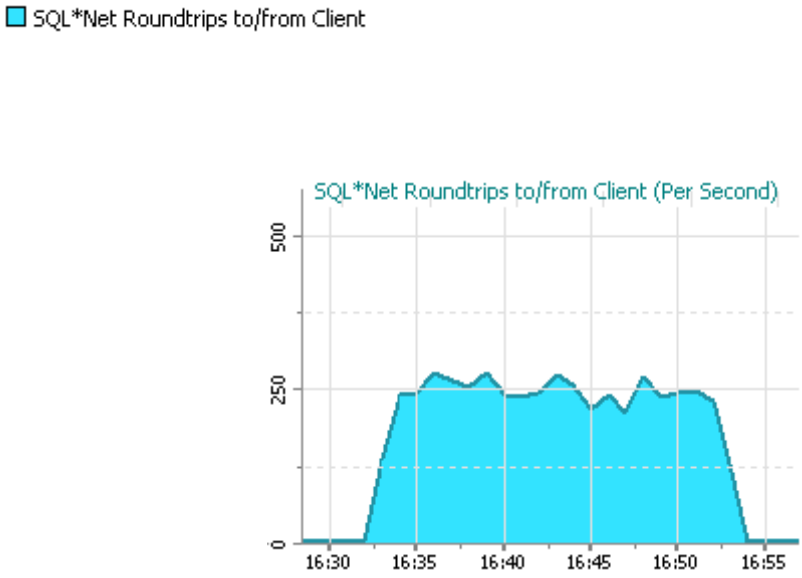

Gráfico XLIII: Peticiones de cliente en la Prueba 2 con auditoría genérica

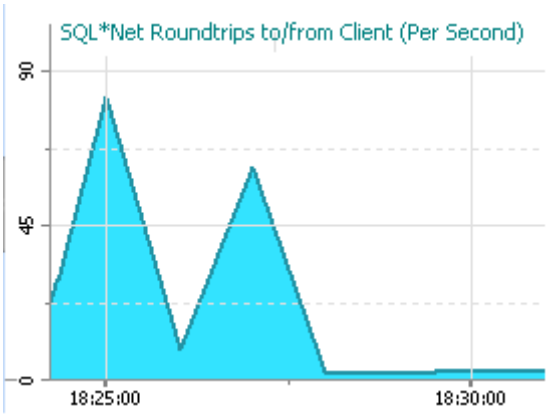

Gráfico XLIV: Peticiones de cliente en la Prueba 6 con auditoría genérica

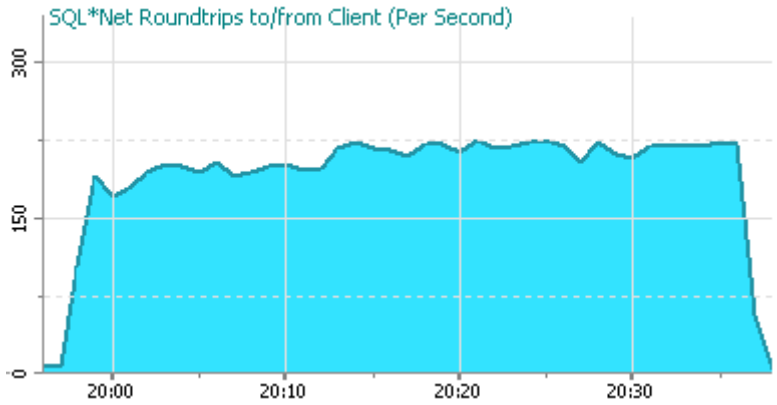

Gráfico XLV: Peticiones de cliente en la Prueba 2 con auditoría de grano fino

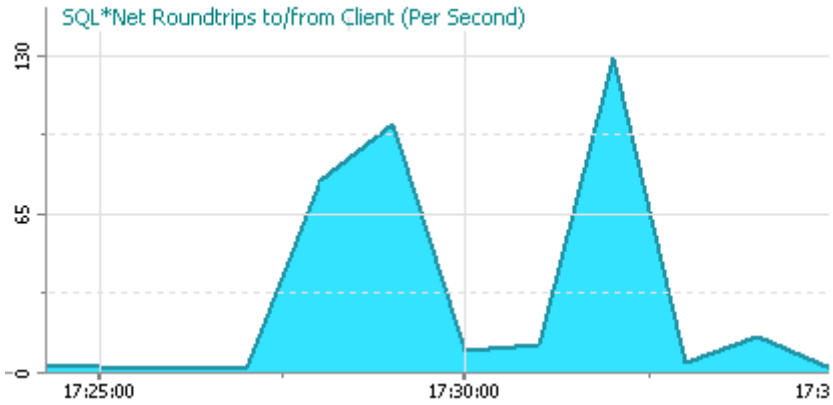

Gráfico XLVI: Peticiones de cliente en la Prueba 6 con auditoría de grano fino

6. Análisis de la recolección

En este punto se ha descrito una infraestructura y unos procedimientos para realizar auditoría de la forma más eficientemente posible. Comenzaremos por describir la infraestructura de auditoría que permita maximizar la eficiencia del servidor de base de datos, siguiendo por la explicación de unos procedimientos para obtener los datos de auditoría: cómo obtener dichos datos y cómo utilizarlos.

6.1. Infraestructura para el análisis de la recolección de los datos

Dado el análisis de rendimiento explicado en el punto **4 rendimiento en la recolección de datos en Oracle 11g**, se ha llegado a una infraestructura que maximiza el rendimiento de una base de datos, siendo ésta una base de datos muy concurrida, donde se realizan operaciones de forma masiva.

De forma general, la infraestructura está recogida de forma resumida en el siguiente esquema:

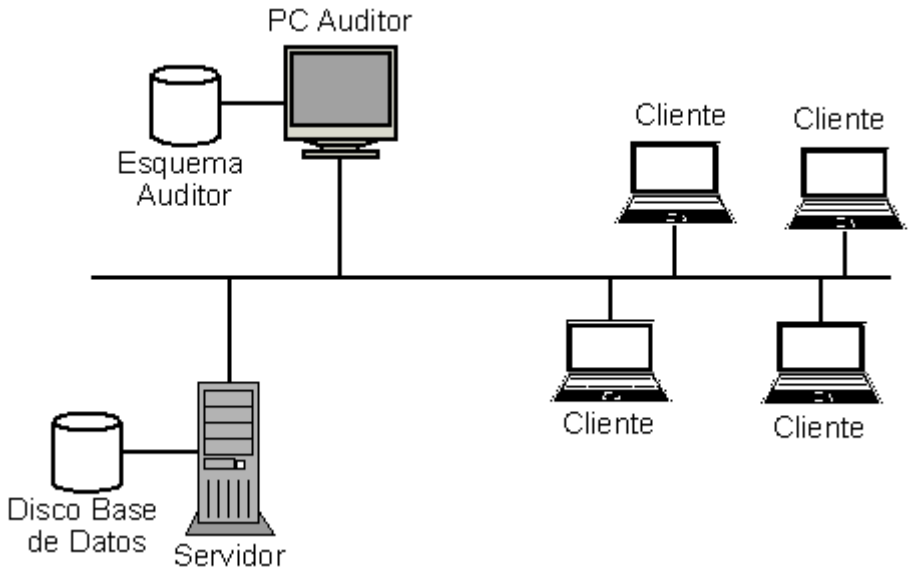

Ilustración 14: Infraestructura recomendada para el análisis de la recolección de los datos de auditoría

En ordenador *Servidor* será el que posea la base de datos que se quiere auditar, y su nombre o su IP (según la infraestructura de la red) serán públicos para los ordenadores clientes con el objetivo de poder realizar operaciones sobre el esquema.

Aparte, habrá un ordenador que contenga instalado otra instancia de base de datos. Este ordenador, llamado en el esquema anterior *PC Auditor*, será el utilizado por el auditor de la base de datos para realizar las operaciones pertinentes con el objetivo de despejar de gasto computacional lo máximo posible al servidor de base de datos. No es conveniente que su dirección IP o su nombre de red sean públicos como con el *Servidor*, pues será de uso único para el auditor.

En la misma red, los clientes serán los ordenadores donde los usuarios puedan acceder a los datos del servidor, cada uno con un nombre de usuario y su contraseña correspondiente, así como los permisos únicamente para realizar las operaciones que el administrador les permita.

Una opción que puede tomar el diseñador de la red es el de poner el *PC Auditor* fuera de la red, conectado directamente con el *Servidor*. Se necesitarían dos tarjetas de red para el *Servidor*: una para la conexión con *PC Auditor* y otra para la conexión con los clientes. En este caso, la red quedaría como se muestra en la siguiente figura:

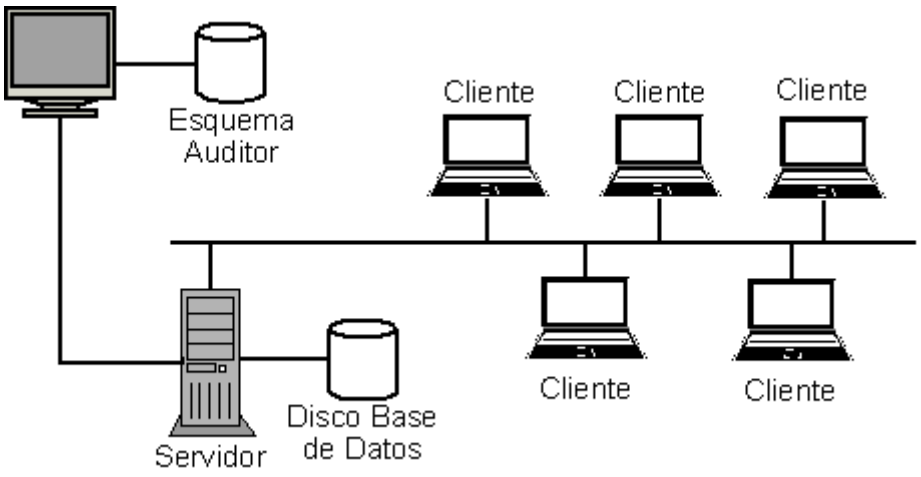

Ilustración 15: Infraestructura alternativa para el análisis de la recolección de los datos de auditoría

La ventaja que tiene esta infraestructura alternativa es el hecho de que las transferencias de datos entre el Servidor y el *PC Auditor* irán más rápido, lo que aligera el trabajo del auditor en caso de ser un trabajo con una transferencia de información muy pesada. La principal desventaja es que es prácticamente inviable para bases de datos distribuidas, encareciendo el coste de la infraestructura y complicando su diseño.

A continuación se describirán más detalladamente los dos ordenadores necesarios para realizar esta tarea.

6.1.1. Servidor

Este ordenador tendrá la instancia de la base de datos que va a ser utilizada por los clientes. Para un funcionamiento óptimo, tendrá un disco donde se recogen los espacios de tablas por defecto, los de auditoría y un espacio de tabla que contendrá el esquema utilizado por los clientes.

El siguiente esquema muestra los espacios de tablas requeridos por la instancia de la base de datos del servidor:

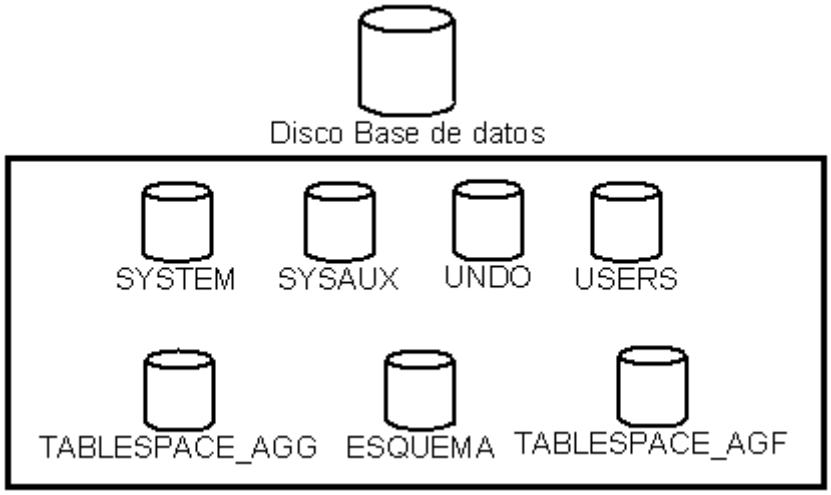

Ilustración 16: Esquema de espacios de tablas del Servidor

El espacio de tablas SYSTEM es creado automáticamente en la instalación, y es usado para gestionar la base de datos, ya que tiene el diccionario de datos. También contiene las tablas y vistas que utiliza el administrador de la base de datos dentro del esquema SYS.

El espacio de tablas SYSAUX está creado como soporte auxiliar al espacio de tablas SYSTEM, y así reducir la carga de éste.

El espacio de tablas UNDO contiene información de tipo “deshacer” para poder utilizar el comando *rollback*.

El espacio de tablas USERS contiene las tablas y los datos permanentes de los usuarios, por defecto. Sin embargo, para tener más seguro el esquema que va a ser utilizado por los usuarios, se creará otro espacio de tablas llamado ESQUEMA para simbolizar y tener más controlado el esquema que es usado por los usuarios.

El espacio de tablas TABLESPACE_AGG contendrá la tabla AUD$, que es la que contiene todos los registros de auditoría genérica, con el objetivo de no sobrecargar el espacio de tablas SYSTEM, que es donde inicialmente está la

tabla AUD$. Para mover esta tabla al espacio de tablas indicado, el administrador de la base de datos deberá ejecutar la siguiente sentencia:

```
ALTER TABLE AUD$ MOVE TABLESPACE TABLESPACE_GGA;
```

El espacio de tablas TABLESPACE_FGA contendrá la tabla FGA_LOG$, que es la que contiene todos los registros de auditoría de grano fino, con el objetivo de no sobrecargar el espacio de tablas SYSTEM, que es donde inicialmente está la tabla FGA_LOG$. Para mover esta tabla al espacio de tablas indicado, el administrador de la base de datos deberá ejecutar las siguientes sentencias:

```
ALTER TABLE FGA_LOG$ MODIFY (PLHOL CLOB);
ALTER TABLE FGA_LOG$ MOVE TABLESPACE TABLESPACE_FGA;
```

La primera sentencia modifica el campo PLHOL, que es de tipo LONG y que impide un cambio de espacio de tabla, a tipo CLOB, que contiene datos de caracteres basados en el juego de caracteres predeterminados del servidor, y su tamaño máximo es de 4 gigabytes, lo que le hace el tipo más indicado para el cambio. La razón es que *Oracle* no permite un cambio de espacio de tabla con el tipo LONG, tan y como se indica en la siguiente sentencia:

```
SQL> ALTER TABLE FGA_LOG$ MOVE TABLESPACE TABLESPACE_FGA;
ALTER TABLE FGA_LOG$ MOVE TABLESPACE TABLESPACE_FGA
*
ERROR en lÝnea 1:
ORA-00997: uso no vßlido del tipo de dato LONG
```

[10]

Además, el servidor tendrá un usuario llamado *auditor_ser* que tendrá los permisos de lectura y borrado sobre las tablas AUD$ y FGA_LOG$ con el fin de poder manipular esas tablas, realizar copias de seguridad y vaciado de dichas tablas.

```
create user auditor_ser identified by auditor_ser_secret;
grant connect, resource to auditor_ser;
grant select, delete on SYS.AUD$;
grant select, delete on SYS.FGA_LOG$;
```

Según los datos de rendimiento obtenidos en el punto 4 Rendimiento en la recolección de datos en Oracle 11g, sería muy recomendable tener otro disco en el que depositar los ficheros Redo, para aumentar la eficiencia del servidor. No obstante, se recomienda que sólo se haga en el caso de ser una base de datos muy concurrente y si se realiza una auditoría de nivel medio o alto, pues con el nivel bajo, la mejora apenas se notaría, se gastarían más recursos económicos y aumentaría la probabilidad de fallo de disco.

6.1.2. PC Auditor

Este ordenador tendrá la instancia de base de datos que va a ser utilizada por el auditor. Tendrá un disco donde se recogen los espacios de tablas por defecto y un espacio de tabla que contendrá el esquema utilizado por el auditor.

El siguiente esquema muestra los espacios de tablas requeridos por la instancia de la base de datos del *PC Auditor*:

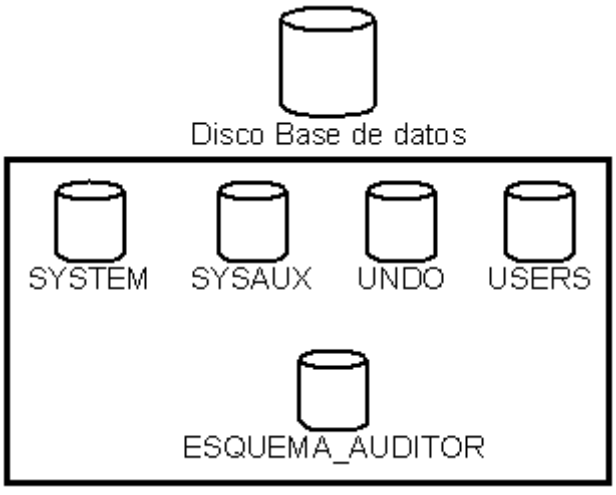

Ilustración 17: Estructura de espacios de tablas del PC Auditor

Los espacios de tablas SYSTEM, SYSAUX, UNDO y USERS están explicados en el punto **6.1.1 Servidor** del presente documento, y tienen la misma configuración, a excepción de la tabla del administrador AUD$, que no será necesaria a menos que se desee auditar las acciones del propio auditor.

El espacio de tablas ESQUEMA_AUDITOR recogerá las tablas necesarias para que el auditor realice su trabajo. El auditor será un usuario llamado *auditor_aud* que tendrá todos los permisos sobre su espacio de tablas asignado. Los permisos mínimos serán los siguientes:

```
create user auditor_aud identified by auditor_aud_secret;
grant connect, resource to auditor_aud;
grant create view to auditor_aud;
grant create table to auditor_aud;
grant create synonym   to auditor_aud;
grant create procedure to auditor_aud;
grant create database link to auditor_aud;
```

Este usuario deberá crear un enlace al ordenador *Servidor* para poder manipular, de forma remota, la tabla AUD$:

```
create public database link enlace_auditoría connect to
auditor_ser identified by auditor_ser_secret using
'BDEmpresa';
```

donde *BDEmpresa* es el nombre de la instancia de la base de datos.

Para que el auditor realice una copia de seguridad de la tabla AUD$, debe copiar dicha tabla en una tabla auxiliar en PC Auditor, según los siguientes pasos:

Paso 1: crear una tabla auxiliar donde guardar los datos de auditoría:

```
create table AUX_AUD_aaaammdd$ AS SELECT * FROM
sys.AUD$@enlace_auditoría where rownum<1;
```

Paso 2: crear un momento concreto fijado para realizar la copia de seguridad:

```
create table hoy (
 hoy date primary key
);
insert into hoy values (sysdate);
```

Paso 3: copiar los datos de auditoría generados hasta el momento fijado:

```
insert into AUX_AUD_aaaammdd$ (SELECT * FROM
sys.AUD$@enlace_auditoría WHERE ntimestamp#<=(select hoy
from hoy));
```

Paso 4: borrar los datos originales copiados:

```
DELETE FROM sys.AUD$@enlace_auditoría where ntimestamp# <=
(select hoy from hoy);
```

Paso 5: realizar copia de seguridad. Al tener nombrada la tabla auxiliar con la fecha actual, el auditor puede tomar la decisión de mantener las tablas en un esquema propio, o realizar una copia de seguridad para ahorrar espacio en disco. Si decide realizar la segunda opción, deberá ejecutar lo siguiente en SQL+ con los permisos adecuados, para crear un directorio donde guardar la copia de seguridad y para poner el espacio de tablas en modo backup:

```
create or replace directory directorio as
'C:\CopiasDeSeguridad';

grant read, write on directory directorio to public;

alter tablespace ESQUEMA_AUDITOR begin backup;
```

Desde la consola de Windows, ejecutar el siguiente comando:

```
expdp auditor_aud/auditor_aud_secret DIRECTORY=directorio
DUMPFILE=AUX_AUD_aaaammdd.dmp LOGFILE=AUX_AUD_aaaammdd.log
TABLES=AUX_AUD_aaaammdd$
```

Por último, cuando termine la realización de la copia de seguridad, deberá ejecutar lo siguiente en SQL+ para que el espacio de tablas deje de estar en modo *backup*:

```
alter tablespace ESQUEMA_AUDITOR end backup;
```

En diferente forma se realizan las tablas auxiliares para la auditoría de grano fino, ya que, como dijimos en el punto **4.6 Rendimiento de backup y recuperación**, es más eficiente realizar una tabla auxiliar en el mismo servidor de la tabla de auditoría de grano fino. Para ello seguiremos los mismos pasos que en auditoría genérica, solo que no hará falta utilizar el enlace '*enlace*_auditoría' porque estaremos dentro del mismo servidor:

Paso 1: crear una tabla auxiliar donde guardar los datos de auditoría:

```
 create table AUX_FGA_LOG_aaaammdd$ AS SELECT * FROM
sys.FGA_LOG$ where rownum<1;
```

Paso 2: crear un momento concreto fijado para realizar la copia de seguridad:

```
create table hoy (
 hoy date primary key
);
insert into hoy values (sysdate);
```

Paso 3: copiar los datos de auditoría generados hasta el momento fijado:

```
insert into AUX_FGA_LOG_aaaammdd$ (SELECT * FROM
sys.FGA_LOG$ WHERE ntimestamp#<=(select hoy from hoy));
```

Paso 4: borrar los datos originales copiados:

```
DELETE FROM sys.FGA_LOG$ where ntimestamp# <= (select hoy
from hoy);
```

Paso 5: realizar copia de seguridad. Al tener nombrada la tabla auxiliar con la fecha actual, el auditor puede tomar la decisión de mantener las tablas en un esquema propio, o realizar una copia de seguridad para ahorrar espacio en

disco. Si decide realizar la segunda opción, deberá ejecutar lo siguiente en SQL+, con los permisos adecuados, para crear un directorio donde guardar la copia de seguridad y para poner el espacio de tablas en modo backup:

```
create or replace directory directorio as
'C:\CopiasDeSeguridad';

grant read, write on directory directorio to public;

alter tablespace ESQUEMA_AUDITOR begin backup;
```

Desde la consola de Windows, ejecutar el siguiente comando:

```
expdp auditor_aud/auditor_aud_secret DIRECTORY=directorio
DUMPFILE=AUX_FGA_LOG_aaaammdd.dmp
LOGFILE=AUX_FGA_LOG_aaaammdd.log
TABLES=AUX_FGA_LOG_aaaammdd$
```

Por último, cuando termine la realización de la copia de seguridad, deberá ejecutar lo siguiente en SQL+ para que el espacio de tablas deje de estar en modo *backup*:

```
alter tablespace ESQUEMA_AUDITOR end backup;
```

NOTA: la nomenclatura *aaaammdd* corresponde a la fecha en la que se realiza la copia de seguridad, siendo *aaaa* el año de realización, *mm* el mes y *dd* el día dentro del mes. Por tanto, una copia de seguridad de la tabla AUD$ realizada el día 14 de marzo del año 2009 se llamará AUX_AUD_20090314$.

6.1.3. Dependencia de la densidad de operaciones

Toda esta infraestructura se simplifica en un solo ordenador cuando la base de datos es poco concurrida. Bastará con un solo ordenador *Servidor* con el mismo esquema de espacios de tablas, además del ESQUEMA_AUDITOR. En este caso, en lugar de utilizar otro ordenador para realizar la copia de seguridad, se utilizará el mismo servidor, sin necesidad de enlaces y con un solo usuario *auditor_ser*, poseedor del espacio de tablas ESQUEMA_AUDITOR y realizando las mismas operaciones, pero en un solo ordenador.

6.2. Búsqueda de un evento en el tiempo con Log Miner

Un evento es una acción realizada sobre la base de datos en un momento concreto por parte del usuario. Desde el punto de vista del auditor puede verse como una sentencia SQL ejecutada por un usuario en un instante determinado y que dio lugar a la inserción de un registro en la tabla de auditoría.

La búsqueda de un evento en el tiempo requiere una herramienta que permita leer los eventos de los ficheros de Redo Log, que guardan el historial de cambios realizados a la base de datos. Cada entrada de los ficheros de Redo Log guarda los cambios realizados, con el objetivo de ser utilizados en las operaciones de recuperación hacia delante (*roll-fordward*) por culpa de una caída o similar.

La herramienta a utilizar se llama *Log Miner*. Esta herramienta permite leer los registros de Redo Log a partir de sus ficheros, mediante consultas a una vista llamada *v$logmnr_contents*. A continuación se describen los campos más importantes de ésta vista:

- TIMESTAMP: indica el momento exacto en que el cambio se realizó.
- COMMIT_TIMESTAMP: momento exacto en que la transacción fue confirmada.
- TABLE_NAME: nombre de la tabla que ha sido modificada, en caso de ser una tabla.
- TABLE_SPACE: nombre del espacio de tablas que contiene el segmento modificado. Si el cambio concierne a varios espacios de tabla, esta columna está vacía.
- USERNAME: nombre del usuario que realizó el cambio.

- OPERATION: operación realizada para el cambio. Sus posibles valores son los siguientes: INSERT, UPDATE, DELETE, DDL (en caso de ser una sentencia de definición de datos), START (inicio de transacción), COMMIT (confirmación de la transacción), ROLLBACK (cambio causado por dehacer una transacción), LOB_WRITE (invocación a la función DBMS_LOB.WRITE), LOB_TRIM (invocación a la función DBMS_LOB.TRIM), LOB_ERASE (invocación a la función DBMS_LOB.ERASE), SELECT_FOR_UPDATE (sentencia SELECT FOR UPDATE), SEL_LOB_LOCATOR (sentencia SELECT que devuelve un LOB), MISSING_SCN (hueco en el registro de Redo), INTERNAL (operación lanzada por la propia base de datos), UNSUPPORTED (operación no soportada por *Log Miner*).

- SQL_REDO: reconstrucción de la sentencia SQL ejecutada.

- SQL_UNDO: reconstrucción de la sentencia SQL que deshace el cambio realizado por la sentencia SQL ejecutada.

Para utilizar *Log Miner* es necesario especificar una ruta donde *Log Miner* pueda ubicar su diccionario de objetos para poder funcionar. Para ello se debe especificar su ubicación el parámetro *utl_file_dir* mediante la siguiente sentencia de ejemplo:

```
alter system set utl_file_dir='C:\DatosMarioPFC\LogMiner'
scope=spfile;
shutdown immediate;
startup;
```

Y tras volver a abrir la base de datos, se debe construir el diccionario mediante la siguiente sentencia ejemplo:

```
exec DBMS_LOGMNR_D.BUILD( DICTIONARY_FILENAME
=>'dictionary.ora', DICTIONARY_LOCATION =>
'c:\DatosMarioPFC\LogMiner');
```

A continuación hay que especificar a *Log Miner* qué ficheros de Redo Log se quieren analizar. Para ver los ficheros Redo Log es necesario usar la siguiente sentencia:

```
SQL> select member from v$logfile;
MEMBER
-------------------------------------------------
C:\DATOSMARIOPFC\BDMARIO\REDO01.LOG
C:\DATOSMARIOPFC\BDMARIO\REDO02.LOG
C:\DATOSMARIOPFC\BDMARIO\REDO03.LOG
```

Para cargar los ficheros y así poderlos analizar, hay que utilizar la siguiente sentencia ejemplo para cada fichero:

```
SQL> exec
DBMS_LOGMNR.add_logfile('C:\DATOSMARIOPFC\BDMARIO\REDO01.L
OG');
Procedimiento PL/SQL terminado correctamente.
```

Una vez preparado todo esto, ya se puede iniciar la sesión *Log Miner* indicando la ubicación del diccionario de *Log Miner*, mediante la sentencia:

```
EXECUTE DBMS_LOGMNR.START_LOGMNR(DICTFILENAME
=>'c:\DatosMarioPFC\LogMiner\dictionary.ora');
```

A partir de este momento ya se pueden realizar consultas a la vista antes mencionada v$logmnr_contents, por ejemplo la siguiente sentencia:

```
select sql_redo, username, timestamp from
v$logmnr_contents where timestamp>sysdate-1;
```

Cuando se acaba de utilizar el *Log Miner*, es necesario terminar la sesión de análisis de los ficheros Redo Log, mediante la siguiente sentencia:

```
EXECUTE DBMS_LOGMNR.END_LOGMNR();
```

Esta forma de buscar eventos permite conocer más datos, como el momento en que la transacción fue confirmada, la sentencia SQL que permite deshacer el

cambio u otros tipos de operaciones que no se pueden auditar por interactuar con LOBs. Sin embargo es más costosa de utilizar, pues para encontrar un evento hay que tener en cuenta que aquí se encuentran todas las operaciones realizadas. Además, si se ejecuta un *rollback*, la transacción se deshace y no queda constancia de su ejecución en la vista *v$logmnr_contents*.

También, para poder utilizar este método se necesita importar la copia de seguridad de la base de datos completa, y es incómoda de utilizar debido a que no presenta las operaciones tal y como se hicieron, sino en un formato interno.

6.3. Obtención de un dato específico de auditoría

Un dato es la información obtenida de un evento auditado. Es un registro de auditoría generado por el evento y que ha sido guardado en la tabla de auditoría, ya sea AUD$ para la auditoría genérica o FGA_LOG$ para la auditoría de grano fino.

La búsqueda de un dato para la auditoría genérica es diferente a la búsqueda de un evento en el tiempo para la auditoría de grano fino. Por ello, se explicarán en apartados diferentes.

A modo de resumen, la búsqueda de un dato implica recuperar todos los backups de las tablas auxiliares donde se guardaron los datos de auditoría. A continuación se creará una vista, diferente para auditoría genérica y de grano fino, para la mejor visualización de los datos. Mediante un conjunto de consultas explicadas en los siguientes puntos, y las vistas creadas, se podrá llegar a obtener los datos deseados.

6.3.1. Búsqueda de un dato para la auditoría genérica

La auditoría genérica tiene por objetivos:

- ver qué operaciones realizó un usuario en un periodo de tiempo.
- ver qué usuarios realizaron un conjunto de operaciones en un periodo de tiempo.

Por tanto, para buscar un dato sobre un evento en el tiempo no es necesario más herramientas que las copias de seguridad de las tablas auxiliares de AUD$, que son las creadas con el nombre AUX_AUD_*aaaammdd*$ y cuya creación se ha explicado en el apartado **6.1.2 PC Auditor** del presente documento.

El auditor sólo deberá recuperar las tablas en la base de datos local que existe en el PC Auditor, ya que se realizó una copia de seguridad lógica.

Gracias a que son tablas con los mismos campos, se podrá realizar una unión (operación *UNION*) de todas las tablas, seleccionando los parámetros requeridos para ello.

Para una mejor visualización de los datos, será muy recomendable utilizar una vista análoga a DBA_AUDIT_TRAIL, que muestra todos los registros de auditoría genérica de todas las fechas necesarias. Para ello será necesario crearla de la siguiente manera:

```
CREATE VIEW VISTA_AUDITORÍA (
     os_username,
     username,
     userhost,
     terminal,
     timestamp,
     owner,
     obj_name,
     action,
     action_name,
     new_owner,
     new_name,
     obj_privilege,
     sys_privilege,
     admin_option,
     grantee,
     audit_option,
```

```
      ses_actions,
      logoff_time,
      logoff_lread,
      logoff_pread,
      logoff_lwrite,
      logoff_dlock,
      comment_text,
      sessionid,
      entryid,
      statementid,
      returncode,
      priv_used,
      client_id,
      econtext_id,
      session_cpu,
      extended_timestamp,
      proxy_sessionid,
      global_uid,
      instance_number,
      os_process,
      transactionid,
      scn,
      sql_bind,
      sql_text,
      obj_edition_name )
AS
select spare1,
       userid,
       userhost,
       terminal,
       cast (
           (from_tz(ntimestamp#,'00:00') at local) as
date),
       obj$creator,
       obj$name,
       aud.action#,
       act.name,
       new$owner,
       new$name,
       decode(aud.action#,
              108, null,
              109, null,
              114, null,
              115, null,
              auth$privileges),
       decode(aud.action#,
              108, spm.name,
              109, spm.name,
              null)
```

```
      decode(aud.action#,
             108, substr(auth$privileges,1,1),
             109, substr(auth$privileges,1,1),
             114, substr(auth$privileges,1,1),
             115, substr(auth$privileges,1,1),
             null),
       auth$grantee,
       decode(aud.action#,
             104,
        aom.name,
             105, aom.name,
             null),
       ses$actions,
       cast((from_tz(cast(logoff$time as
timestamp),'00:00') at local) as date),
       logoff$lread,
       logoff$pread,
       logoff$lwrite,
       decode(aud.action#,
             104, null,
             105, null,
             108, null,
             109, null,
             114, null,
             115, null,
             aud.logoff$dead),
       comment$text,
       sessionid,
       entryid,
       statement,
       returncode,
       spx.name,
       clientid,
       auditid,
       sessioncpu,
       from_tz(ntimestamp#,'00:00') at local,
       proxy$sid,
       user$guid,
       instance#,
       process#,
       xid,
       scn,
       to_nchar(substr(sqlbind,1,2000)),
       to_nchar(substr(sqltext,1,2000)),
       obj$edition
from
(
AUX_AUD_aaa1m1d1
UNION
```

```
AUX_AUD_aaa2m2d2
UNION
…
AUX_AUD_aaaNmNdN
) aud, system_privilege_map@enlace_auditoría spm,
system_privilege_map@enlace_auditoría spx,
     STMT_AUDIT_OPTION_MAP@enlace_auditoría aom,
audit_actions@enlace_auditoría act
where   aud.action#     = act.action    (+)
  and - aud.logoff$dead = spm.privilege (+)
  and   aud.logoff$dead = aom.option#   (+)
  and - aud.priv$used   = spx.privilege (+);
```

Tras la construcción de esta vista, el auditor podrá visualizar con mayor facilidad los eventos realizados por el usuario. Los campos más importantes de la vista, y los que son más recomendables de visualizar, son los siguientes:

- El campo USERNAME muestra el usuario que realizó la operación.
- El campo USERHOST muestra el nombre del ordenador desde el que se realizó la operación.
- El campo TIMESTAMP muestra el momento exacto en que se realizó la operación.
- El campo OWNER muestra el propietario del objeto sobre el que el usuario realizó la operación.
- El campo OBJ_NAME muestra el objeto sobre el que el usuario realizó la operación.
- ACTION_NAME: muestra SESSION REC en caso de ser auditoría *by session*, o la operación realizada, en caso de ser auditoría *by access*.
- Los campos NEW_OWNER y NEW_NAME muestran el nuevo propietario y el nuevo nombre del objeto en el caso de que la operación haya sido de renombrado o de creación.

- El campo OBJ_PRIVILEGE muestra el privilegio de objeto que se ha manejado en caso de que la operación sea GRANT o REVOKE.
- El campo SYS_PRIVILEGE muestra el privilegio de sistema que se ha manejado en caso de que la operación sea GRANT o REVOKE.
- El campo GRANTEE muestra el usuario al que se le ha dado o quitado un privilegio en caso de que la operación sea GRANT o REVOKE.
- El campo LOGOFF_TIME muestra el momento exacto en el que el usuario termina su sesión, en caso de que la operación sea LOGOFF.
- El campo PRIV_USED muestra el privilegio usado para realizar la operación.
- El campo EXTENDED_TIMESTAMP muestra el momento exacto en el que el usuario inicia su sesión en la que ejecuta el evento.

El resto de campos pueden ser eliminados de la vista o no, según los intereses del auditor. A continuación se mostrarán dichos campos:

- OS_USERNAME: Nombre de usuario del sistema operative que realizó la operación.
- TERMINAL: Identificador del terminal de usuario.
- ACTION: Es el código numérico de la operación realzada.
- ADMIN_OPTION: Indica si un rol o u sys_priv fue creado con la opción ADMIN (A/-)
- AUDIT_OPTION: Indica el conjunto de opciones de auditoría de la sentencia de auditoría.

- SES_ACTIONS: Es el resumen de sesión, en caso de ser auditoría *by session*, dejándose vacía en caso de ser auditoría *by access*, pues en ese caso la operación quedaría grabada en el campo ACTION_NAME. Es una cadena de 12 caracteres, uno por cada tipo de acción, en este orden: Alter, Audit, Comment, Delete, Grant, Index, Insert, Lock, Rename, Select, Update, Flashback. Valores: "-" = ninguno, "S" = éxito en la operación, "F" = no éxito en la operación.
- LOGOFF_LREAD: Lecturas lógicas en la sesión.
- LOGOFF_PREAD: Lecturas físicas en la sesión.
- LOGOFF_LWRITE: Escrituras lógicas en la sesión.
- LOGOFF_DLOCK: Bloqueos detectados en la sesión.
- COMMENT_TEXT: Comentario sobre la operación realizada.
- SESSIONID: Identificador numérico para cada sesión Oracle.
- ENTRYID: Identificador numérico del registro de auditoría en la sesión.
- STATEMENTID: Identificador numérico de la sentencia ejecutada.
- RETURNCODE: Error Oracle generado por la acción. Cero en caso de éxito de la acción.
- PRIV_USED: Privilegio usado para ejecutar la acción.
- CLIENT_ID: Identificador del cliente en la sesión.

- ECONTEXT_ID: Identificador del contexto de ejecución para cada acción.
- SESSION_CPU: Tiempo de CPU usada en la sesión.
- PROXY_SESSIONID: Número serial de la sesión del proxy, si se usa un proxy.
- GLOBAL_UID: Identificador global del usuario.
- INSTANCE_NUMBER: Número de instancia del parámetro de inicialización del fichero 'init.ora'.
- OS_PROCESS: Description of DBA_AUDITIdentificador del proceso del sistema operatvo.
- TRANSACTIONID: Identificador de la transacción.
- SCN: Número SCN (System Change Number) de la sentecia.
- SQL_BIND: Datos de la variable BIND de la sentencia ejecutada.
- OBJ_EDITION_NAME: Edición del objeto auditado.
- SQL_TEXT: no muestra nada.

[11]

6.3.2. Búsqueda de un dato para la auditoría de grano fino

La filosofía de la auditoría de grano fino cambia debido a que necesitamos saber:

- Qué información se vio en una consulta.

- Qué información se insertó en una inserción.
- Qué información se borró en un borrado.
- Qué información fue cambiada por qué información.

Para ello dividiremos las cuatro operaciones en dos grupos: el grupo que necesita de las copias de seguridad, en las que deben ser incluidos los ficheros redo, para ver qué información se vio, se borró o fue cambiada por qué información; y el grupo, que sólo incluye la inserción, que no necesita el apoyo de las copias de seguridad.

En el primer grupo hay que realizar una importación de la base de datos completa. Para ello hay que crear una base de datos nueva en PC Auditor. Seguidamente hay que crear los espacios de tabla de la nueva base, haciéndolos coincidir con los nombres de los espacios de tablas de la base de datos de origen. Luego, en la línea de comandos hay que ejecutar la siguiente sentencia:

```
imp userid=system/system_secret
file=C:/bakups/backup20090314.dmp log=imp.log full=y
```

siendo la fecha inmediatamente posterior a la fecha en la que queremos ver qué se hizo.

Una vez recuperada la base de datos, el auditor debe conectarse como administrador:

```
connect / as sysdba;
```

A continuación debe bajar la instancia y montarla sin ser abierta, para recuperar a base de datos en un momento determinado:

```
Shutdown immediate;
Startup mount;
```

Se recupera la base de datos en la fecha deseada con la siguiente sentencia:

```
recover database until time '2009-02-01:16:00:00';
```

A continuación se debe abrir la base de datos con la opción *resetlogs* para realizar una recuperación parcial, pues no queremos recuperar la base de datos hasta el punto actual, sino hasta el punto indicado:

```
alter database open resetlogs;
```

A partir de este momento, hay tres caminos:

- En el caso de que la operación sea una consulta, se ejecuta la consulta auditada para comprobar que es lo que el usuario vio.

- En el caso de ser un borrado, debe cambiarse la operación por una consulta de manera que se visualicen los campos deseados (*select * from tabla where*…) copiando la cláusula *where* de la nueva consulta por la cláusula *where* de la operación de borrado de forma que se visualice qué es lo que el usuario borró.

- En el caso de ser una modificación, se realiza la misma operación que en el caso de ser un borrado, para ver qué información fue cambiada; a continuación se ejecutará la acción de modificación y se volverá a realizar la operación que en el caso de borrado para ver qué información resultó. Con ello tendremos la información de antes y de después de realizar la modificación.

En cuanto a la inserción, bastaría con visualizar la sentencia SQL para ver qué información insertó el usuario.

Para ver las sentencias SQL que fueron ejecutadas es muy recomendable usar una vista análoga a la vista DBA_FGA_AUDIT_TRAIL, que permite visualizar sencillamente los registros de auditoría de grano fino. Para crear dicha vista hay que recurrir a la siguiente sentencia:

```
CREATE VIEW VISTA_AUDITORÍA_FGA (
     session_id,
     timestamp,
     db_user,
     os_user,
     userhost,
     client_id,
     econtext_id,
     ext_name,
     object_schema,
     object_name,
     policy_name,
     scn,
     sql_text,
     sql_bind,
     comment$text,
     statement_type,
     extended_timestamp,
     proxy_sessionid,
     global_uid,
     instance_number,
     os_process,
     transactionid,
     statementid,
     entryid,
     obj_edition_name )
AS
select
      sessionid,
      CAST (
        (FROM_TZ(ntimestamp#,'00:00') AT LOCAL) AS DATE
      ),
      dbuid, osuid, oshst, clientid, auditid, extid,
      obj$schema,      obj$name,      policyname,      scn,
to_nchar(substr(lsqltext,1,2000)),
      to_nchar(substr(lsqlbind,1,2000)), comment$text,
      DECODE(stmt_type,
              1, 'SELECT', 2, 'INSERT', 4, 'UPDATE', 8,
'DELETE', 'INVALID'),
      FROM_TZ(ntimestamp#,'00:00') AT LOCAL,
      proxy$sid, user$guid, instance#, process#,
      xid, statement, entryid, obj$edition
from (
  AUX_FGA_LOG_aaa1m1d1$
  UNION
  …
  UNION
  AUX_FGA_LOG_aaanmndn$
);
```

Donde AUX_FGA_LOG_aaaXmXdX$ son las tablas resultado de la copia de seguridad de la tabla de auditoría de grano fino.

Los campos más importantes son los siguientes:

- El campo TIMESTAMP muestra el momento concreto en el que fue ejecutada la sentencia SQL.
- El campo DB_USER indica el nombre del usuario que ejecutó la sentencia.
- El campo USERHOST muestra el nombre del equipo desde el que el usuario realizó la operación.
- El campo OBJECT_NAME muestra el nombre del objeto sobre el que el usuario realizó la operación.
- El campo POLICY_NAME indica el nombre de la política de auditoría que guardó el registro de auditoría.
- El campo SQL_TEXT muestra la sentencia SQL que fue ejecutada.
- El campo STATEMENT_TYPE indica el tipo de operación de la sentencia.

El resto de campos pueden ser eliminados de la vista o no, según los intereses del auditor. A continuación se mostrarán dichos campos:

- SESSION_ID: Identificador de sesión en la que se ejecutó la sentencia.
- OS_USER: Nombre de usuario del sistema operativo que ejecutó la sentencia
- CLIENT_ID: Identificador de cliente en la sesión Oracle.

- ECONTEXT_ID: Identificador de contexto de ejecución de la acción
- EXT_NAME: Nombre externo.
- OBJECT_SCHEMA: Propietario de la tabla o vista sobre la que se realizó la operación.
- SCN: Número SCN de la sentencia.
- SQL_BIND: Variable BIND de la sentencia.
- COMMENT$TEXT: Comentarios.
- EXTENDED_TIMESTAMP: Momento en el que se ejecutó la sentencia dentro de una zona horaria.
- PROXY_SESSIONID: Número de serie de la sesión de proxy, en caso de usar dicho mecanismo.
- GLOBAL_UID: Identificador global del usuario.
- INSTANCE_NUMBER: Número de instancia del parámetro de inicialización del fichero 'init.ora'.
- OS_PROCESS: Identificador del proceso del sistema operativo.
- TRANSACTIONID: Identificador de la transacción en la que se ejecutó la acción.
- STATEMENTID: Identificador numérico de la sentencia.
- ENTRYID: Identificador numérico del registro de auditoría.
- OBJ_EDITION_NAME: Edición del objeto auditado.

[12]

6.4. Obtención de un conjunto de datos de auditoría

Un conjunto de datos es el resultado de una pregunta concreta de un auditor. Las tres preguntas más habituales que se puede hacer un auditor son:

- ¿Quién hizo la operación *O* entre el momento T_1 y el momento T_2?
- ¿Qué operaciones realizó el usuario *U* entre el momento T_1 y el momento T_2?
- ¿Cuántas operaciones de un mismo tipo han sido realizadas por parte de un usuario determinado? ¿Y desde un mismo Terminal? Esto sirve, por ejemplo, para ver cuántos cambios de salario ha tenido un empleado en el último mes, y cuales han sido esos cambios; detectando así inserciones masivas o borrados masivos.

Para lograr obtener información para contestar a estas tres preguntas, es necesario utilizar las vistas creadas en el punto **6.2 Búsqueda de un evento en el tiempo** del presente documento. Jugando con sus campos, se puede obtener la información básica necesaria, como veremos en los siguientes puntos.

6.4.1. Obtener un conjunto de datos con auditoría genérica

Para la auditoría genérica se utilizará la vista creada en el punto **6.2.1 Búsqueda de un evento en el tiempo para la auditoría genérica** del presente documento, llamada VISTA_AUDTORIA, donde los campos más importantes son: ACTION_NAME, OBJ_NAME, USERNAME, TERMINAL y TIMESTAMP.

Para conseguir información acerca de la primera pregunta, se utilizarán los campos TIMESTAMP y ACTION_NAME. El siguiente ejemplo muestra las operaciones sobre inserción en un periodo determinado:

```
Select * from VISTA_AUDITORÍA where
TIMESTAMP > ‘01/01/2009’ and TIMESTAMP < ‘02/01/2009’
and ACTION_NAME = ‘INSERT’;
```

Para conseguir información acerca de la segunda pregunta, se utilizarán los campos TIMESTAMP y USERNAME. El siguiente ejemplo muestra las operaciones realizadas por un usuario en un periodo determinado:

```
Select * from VISTA_AUDITORÍA where
TIMESTAMP > ‘01/01/2009’ and TIMESTAMP < ‘02/01/2009’
and USERNAME = ‘USUARIO;’
```

La tercera cuestión es la más compleja, pues requiere una cierta elaboración. Para obtener la información es necesario unas consultas a la vista VISTA_AUDITORÍA, como se muestran en los siguientes ejemplos.

El primero muestra la cuenta de operaciones por tipo que ha realizado un usuario en determinadas fechas:

```
Select ACTION_NAME, count(*) from VISTA_AUDITORÍA where
TIMESTAMP > ‘01/01/2009’ and TIMESTAMP < ‘02/01/2009’
and USERNAME = ‘USUARIO’
group by ACTION_NAME;
```

El Segundo ejemplo muestra lo mismo, pero desde un mismo Terminal, algo que debería ser motivo de sospecha:

```
Select ACTION_NAME, count(*) from VISTA_AUDITORÍA where
TIMESTAMP > ‘01/01/2009’ and TIMESTAMP < ‘02/01/2009’
and TERMINAL = ‘NOMBRE_TERMINAL’
group by ACTION_NAME;
```

6.4.2. Obtener un conjunto de datos en auditoría de grano fino

Para la auditoría genérica se utilizará la vista creada en el punto **6.2.2 Búsqueda de un evento en el tiempo para la auditoría de grano fino** del presente documento, llamada VISTA_AUDTORIA_FGA, donde los campos más importantes son: STATEMENT_TYPE, OBJECT_NAME, DB_USER, USERHOST, TIMESTAMP y POLICY_NAME.

Para conseguir información acerca de la primera pregunta, se utilizarán los campos TIMESTAMP y STATEMENT_TYPE. El siguiente ejemplo muestra las operaciones sobre inserción en un periodo determinado:

```
Select * from VISTA_AUDITORÍA_FGA where
TIMESTAMP > '01/01/2009' and TIMESTAMP < '02/01/2009'
and STATEMENT_TYPE = 'INSERT';
```

También es muy útil realizar esta operación, en lugar de por tipo de operación, por política de auditoría, en cuyo caso utilizaríamos:

```
Select * from VISTA_AUDITORÍA_FGA where
TIMESTAMP > '01/01/2009' and TIMESTAMP < '02/01/2009'
and POLICY_NAME = 'politica_insercion';
```

Para conseguir información acerca de la segunda pregunta, se utilizarán los campos TIMESTAMP y DB_USER. El siguiente ejemplo muestra las operaciones realizadas por un usuario en un periodo determinado:

```
Select * from VISTA_AUDITORÍA_FGA where
TIMESTAMP > '01/01/2009' and TIMESTAMP < '02/01/2009'
and DB_USER = 'USUARIO';
```

La tercera cuestión es, como en auditoría genérica, la más compleja. Sería muy recomendable que este tipo de cuestiones estuviesen vigiladas por un procedimiento almacenado, ejecutado tal y como se explica en el punto **4.2 Auditoría de grano fino** del presente documento. Sin embargo, para obtener la

información es necesario desarrollar unas consultas hacia la vista VISTA_AUDITORÍA_FGA, como se muestran en los siguientes ejemplos.

El primero muestra la cuenta de operaciones por tipo que ha realizado un usuario en determinadas fechas:

```
Select count(*) from VISTA_AUDITORÍA_FGA where
TIMESTAMP > '01/01/2009' and TIMESTAMP < '02/01/2009'
and DB_USER = 'USUARIO'
group by STATEMENT_TYPE;
```

El Segundo ejemplo muestra lo mismo, pero desde un mismo Terminal, algo que debería ser motivo de sospecha:

```
Select count(*) from VISTA_AUDITORÍA_FGA where
TIMESTAMP > '01/01/2009' and TIMESTAMP < '02/01/2009'
and USERHOST = 'NOMBRE_TERMINAL'
group by STATEMENT_TYPE;
```

Igualmente, es muy útil realizar estas consultas, en lugar de por operación, por política de auditoría, en cuyo caso se utilizará el campo POLICY_NAME en la agrupación:

```
Select count(*) from VISTA_AUDITORÍA_FGA where
TIMESTAMP > '01/01/2009' and TIMESTAMP < '02/01/2009'
and DB_USER = 'USUARIO'
group by POLICY_NAME;
```

y

```
Select count(*) from VISTA_AUDITORÍA_FGA where
TIMESTAMP > '01/01/2009' and TIMESTAMP < '02/01/2009'
and USERHOST = 'NOMBRE_TERMINAL'
group by POLICY_NAME;
```

respectivamente.

7. Normas para auditar elementos de una organización

En este punto se describirán un conjunto de niveles de auditoría, en qué consisten y qué abarcan. Seguidamente se mostrarán de sus ventajas e inconvenientes e inconvenientes encontrados.

7.1. Niveles de auditoría

Se pueden recoger dos categorías; en cada categoría, varios niveles:

- Categoría de datos personales
- Categoría de datos sensibles para la organización

Como se verá en los siguientes puntos, la categoría de los datos personales es un conjunto que está dentro de la categoría de los datos sensibles para la organización. Se ha elegido realizar un conjunto dentro de una ampliación debido a que se consideran dos tipos de información, según el tipo de ataques que puedan recibir.

En la categoría de datos personales que no son sensibles para la organización, la operación más delicada que puede recibir es la consulta, pues está la Ley Orgánica de Protección de Datos defendiéndolos. Sin embargo, un cambio en esos datos no afectan a los intereses económicos, a los objetivos ni a las funciones de la organización, a excepción de la aplicación de dicha ley con las multas que pueda recibir y la mala imagen que pueda dar.

En la categoría de datos sensibles para la organización se incluyen, además de los anteriores, los datos que pueden afectar a los intereses económicos, a los objetivos y a las funciones de la organización, además de los datos analizados en la parte de *Business Intelligence*.

7.1.1. Categoría de datos personales

En esta categoría, que comprenden todos los datos de carácter personal no sensibles para la organización, se ha establecido una jerarquía de tres niveles, cada uno con sus requisitos, que se explican a continuación:

7.1.1.1. Nivel básico

Este nivel corresponde al nivel de seguridad más bajo. Es muy recomendable que cualquier base de datos cuente, como mínimo, con este nivel, pues cubre los aspectos más básicos de seguridad. Sus requisitos son:

- Se debe tener un Documento de Seguridad, ya sea en papel, en formato electrónico o en formato relacional, en el que queden reflejados qué elementos y qué operaciones sobre dichos elementos pueden realizar los diferentes tipos de usuarios, por roles y/o por usuarios particulares, según la conveniencia de la organización (ver artículo 11.1 de la LOPD [3]).

- Se debe restringir el acceso a los datos por parte de usuarios que no tengan los permisos pertinentes, según el Documento de Seguridad, ya sea mediante privilegios Oracle o mediante otros mecanismos. Es tarea del administrador la gestión de la seguridad, como concepto descrito en el punto **2.2 Necesidad de auditoría** del presente documento.

- Se debe controlar el acceso al sistema, por medio de auditoría genérica, auditando dicho acceso y controlando el número de intentos. Es la auditoría mínima que se debería hacer para controlar el acceso del personal. Teniendo esta información y los horarios de trabajo de los usuarios, se podrán detectar fácilmente anomalías mediante alertas y procedimientos almacenados.

- Se debe tener un repositorio activado, donde guardar los registros de auditoría. Estos son AUD$ y FGA_LOG$ para auditoría genérica y de grano fino, respectivamente. Para tener activado el repositorio de auditoría genérica, será necesario tener el valor *DB* en el parámetro de la base de datos *audit_trail* tal y como se explica en el punto **2.2.1.1 Recolección de datos en auditoría general** del presente documeto.

- Guardar registros de auditoría durante al menos 1 años (ver artículo 47.1 de la LOPD [3]).

7.1.1.2. Nivel medio

El nivel medio está diseñado para organizaciones medianas y grandes, que manejan datos de importancia y cuya seguridad debe ser más fuerte. Sus requisitos son:

- Cumplir el nivel básico.

- Se debe designar un responsable de seguridad (ver artículo 16 de la LOPD [3]) distinto del administrador, con un ordenador distinto del servidor para realizar sus funciones, que son:

    - Evolucionar las políticas de auditoría.

    - Obtener información sobre hechos excepcionales.

    - Obtener información sobre datos de auditoría, tal y como se explica en los puntos **6.4 Obtención de un dato** y **6.5 Obtención de un conjunto de datos** del presente documento.

    - Realización de copias de seguridad y limpieza de las tablas de auditoría.

- Se debe auditar y vigilar los intentos de realización de operaciones, autorizadas o no, sobre objetos de la base de datos y/o sobre los datos personales con el objetivo de limitar dichos intentos por parte de un usuario. Esto se podrá realizar con auditoria genérica.

- Se debe auditar los intentos fallidos de conexión del usuario auditor, único usuario que puede ver el contenido de las tablas de auditoría, usando el envío de mensajes tanto al propio auditor como al administrador de la base de datos cada vez que esto ocurra. Esto se podrá realizar con auditoria genérica.

- Separar los ficheros Redo en otro disco dentro del servidor de base de datos para obtener mayor eficiencia, en el caso de ser una base de datos muy concurrente.

- Guardar registros de auditoría durante al menos 3 años (ver artículo 47.1 de la LOPD [3]).

7.1.1.3. Nivel alto

El nivel alto es de utilidad para grandes organizaciones, donde la información es su valor más importante. Y no sólo datos de la organización, sino también datos personales, datos de mercado o datos económicos. Sus requisitos son:

- Cumplir el nivel medio.

- Copias de seguridad guardadas en ubicaciones diferentes.

- Repositorio de tablas de auditoría en servidor externo al de la base de datos.

- Desarrollo de alertas para la detección de posibles irregularidades: procedimientos almacenados que vigilan los movimientos que el auditor cree más conveniente vigilar. Esto se podrá realizar con auditoria de grano fino.

Al guardar los datos de auditoría en un repositorio externo, la infraestructura se torna algo más compleja, pero a su vez más segura. Quedaría aproximadamente como se puede ver en la siguiente ilustración:

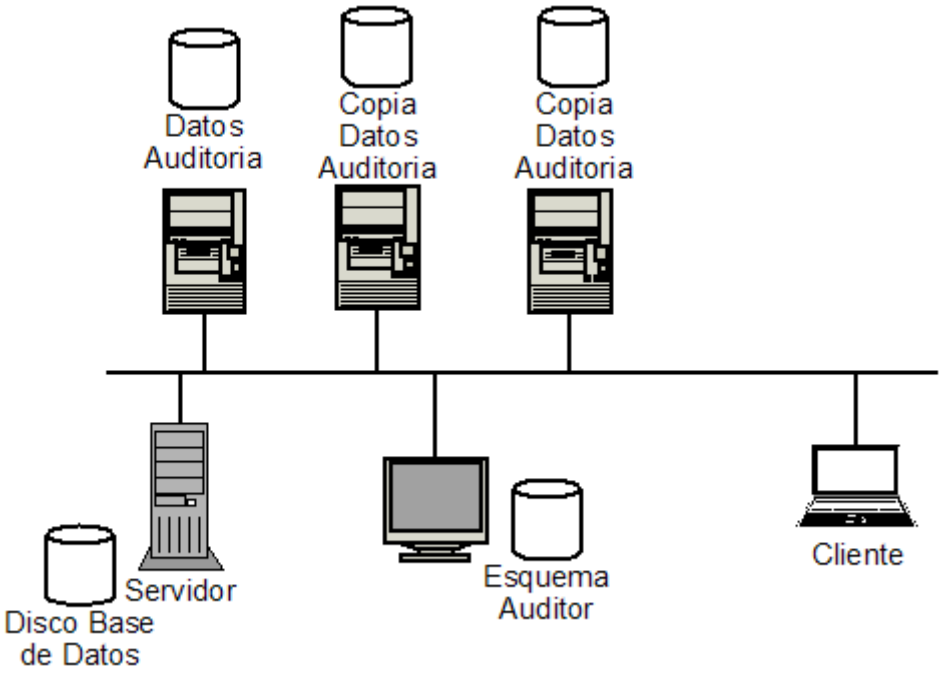

Ilustración 18: Infraestructura recomendada para el análisis de la recolección de los datos de auditoría de nivel alto

Como infraestructura alternativa, al igual que describimos en el punto **5.1 Infraestructura para el análisis de la recolección de los datos**, se puede proponer la descrita en la siguiente figura para la mejora del rendimiento de la base de datos:

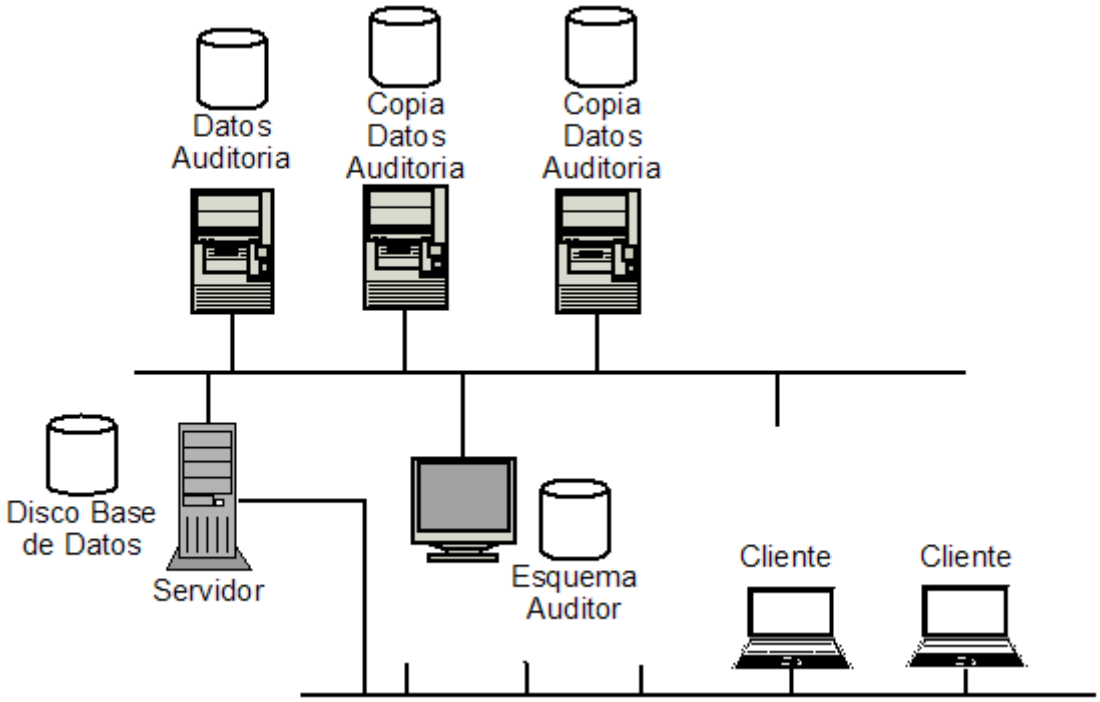

Ilustración 19: Infraestructura alternativa para el análisis de la recolección de los datos de auditoría de nivel alto

7.1.2. Categoría de datos sensibles para la empresa

En esta categoría, que comprenden, además de los datos personales, todos los datos sensibles para la organización ya sean de carácter personal o no, se ha establecido igualmente una jerarquía de tres niveles, cada uno con sus requisitos, que se explican a continuación:

7.1.2.1. Nivel básico

Este nivel es análogo al nivel básico de la categoría de datos personales, explicado en el punto **7.1.1.1 Nivel básico** del presente documento.

7.1.2.2. Nivel medio

Este nivel es análogo al nivel medio de la categoría de datos personales, explicado en el punto **7.1.1.2 Nivel medio** del presente documento, más las siguientes características:

- Se deben auditar los intentos de actualización de los datos personales de los usuarios sensibles para la organización por parte de los propios usuarios. Cada vez que se actualicen dichos datos con una frecuencia mayor de la esperada, el auditor deberá investigar este hecho sobre quién ha realizado las modificaciones, desde qué ordenador y cuándo, por medio de procedimientos almacenados, con el objetivo de detectar irregularidades dentro de la plantilla de la organización. Esto se podrá realizar con auditoria de grano fino.

- Se deben auditar los intentos de actualización y borrado de los datos sensibles para la organización por parte de los usuarios, ya sean autorizados o no autorizados. Cada vez que se actualicen dichos datos, sin frecuencia límite, el auditor deberá investigar este hecho sobre quién ha realizado las modificaciones, desde qué ordenador y cuándo, por medio de procedimientos almacenados, con el objetivo de detectar irregularidades dentro de la plantilla de la organización. Esto se podrá realizar con auditoria de grano fino.

7.1.2.3. Nivel alto

Este nivel es análogo al nivel alto de la categoría de datos personales, explicado en el punto **7.1.1.3 Nivel alto** del presente documento.

7.2. Ventajas y desventajas de los niveles de auditoría

A continuación se enumerarán un conjunto de ventajas e inconvenientes de los niveles de auditoría expuestos en el punto anterior **7.1 Niveles de auditoría** del presente documento, nivel por nivel.

Las ventajas e inconvenientes del nivel básico están incluidas en el nivel medio, y las ventajas e inconvenientes del nivel medio están incluidas dentro del nivel alto.

7.2.1. Nivel básico

7.2.1.1. Ventajas

Una de las principales ventajas que tiene este nivel es la documentación. Es de gran importancia este hecho debido a que facilita la detección de errores de asignación en los permisos y la evolución de éstos con los continuos cambios.

La gestión de privilegios permite al administrador y al auditor constar en el sistema quién tiene acceso y quién no tiene acceso a los objetos, es decir, quién tiene derecho a ejecutar una sentencia SQL o el acceso a un objeto de otro usuario. En Oracle, hay dos tipos de privilegios: privilegios del sistema y privilegios de objeto. Los privilegios pueden ser asignados a un usuario o un rol. La gestión de privilegios permite tener controlado todo ello.

El control de acceso a la base de datos permite saber quién está y quién no está conectado en este momento, así como quién estuvo y quién no estuvo conectado a la base de datos en un momento determinado. Esto permite saber si un usuario estaba o no estaba conectado en el momento en que se produjo algún error o vulnerabilidad.

Además, permite detectar infracciones del tipo leve, descritas en el artículo 44.2 de la LOPD [3].

7.2.1.2. Desventajas

La desventaja principal de este nivel es en una base de datos con altos niveles de conexiones y desconexiones, en cuyo caso se puede descontrolar con cierto nivel dicho control.

Aparece la gestión de repositorios: las tablas de audtoría. El administrador/auditor debe gestionar dichos repositorios si ocurren fallos como llenado de espacios de tabla u otro tipo de inconvenientes.

7.2.2. Nivel medio

7.2.2.1. Ventajas

Con respecto al nivel básico, se obtiene un control casi absoluto de las operaciones por parte de los usuarios. Cada operación que el auditor considera importante es auditada en las tablas de auditoría.

Además, hay una persona que gestiona todos estos datos, pudiendo detectar otras vulnerabilidades nuevas dentro del sistema mediante el estudio de la documentación y la evolución de ésta.

Además, permite detectar infracciones del tipo grave, descritas en el artículo 44.3 de la LOPD [3], e infracciones del tipo muy grave, descritas en el artículo 44.4 de la LOPD [3].

Por último, mejora el rendimiento del servidor debido a que se separan los ficheros Redo en discos diferentes.

7.2.2.2. Desventajas

La principal desventaja de este nivel es el gasto económico extra en personal y en infraestructura de auditoría. Por ello, este nivel está planteado para ser rentable a organizaciones con un nivel medio, alto o muy alto de flujo de información.

7.2.3. Nivel alto

7.2.3.1. Ventajas

La ventaja añadida del nivel alto con respecto al nivel medio es que facilita la conservación de los datos de auditoría en caso de pérdida de una de las copias, pudiendo ser dichos datos recuperados.

Además, el hecho de externalizar las tablas de auditoría mejora el rendimiento del servidor de base de datos, a no ser que el flujo de información entre el servidor de la base de datos y los clientes sea tan alto que aumentar el flujo de información en la red con el servidor de las tablas de auditoría haga que se ralentice la red, lo que se convertiría en desventaja.

Sin embargo, al externalizar las tablas de auditoría estamos facilitando la labor al auditor en caso de ser una base de datos distribuida, pues tendría toda la información de auditoría en un solo repositorio.

Por otro lado, se tiene un control de eventos sospechosos. Operaciones que pueden hacer vulnerable el sistema son detectadas gracias a los elementos de auditoría como políticas de auditoría y procedimientos almacenados.

Por último, el uso de múltiples alertas mejora la seguridad, ya que en caso de posible ataque, su detección sería prácticamente inmediata.

7.2.3.2. Desventajas

Puede ser una desventaja el gasto extra en infraestructura. Sin embargo, las empresas con un nivel alto o muy alto de información poseen, normalmente, diferentes servidores, incluso en diferentes edificios, por lo que el gasto económico disminuye. No es así con el gasto computacional.

En cuanto al uso de alertas, para su creación hay dos caminos: la creación de alertas de forma manual, que es costoso en tiempo y personal; y la creación de alertas usando herramientas de auditoría, con un coste económico relativamente alto. Dos de estas herramientas están descritas en el apartado **4.3.2 Herramientas de auditoría** del presente documento.

8. Conclusiones y líneas futuras

La labor del auditor informático tiene una cierta complicación a la hora de comenzar su realización. El rendimiento del sistema varía en función del nivel de seguridad. El rendimiento y la seguridad deben estar equilibrados, y por ello es necesario llegar a una arquitectura del sistema que nos permita mantener segura la base de datos maximizando su disponibilidad y eficiencia.

Existen metodologías que permiten facilitar la labor del auditor. Estas metodologías están basadas en la experiencia del auditor y, por tanto, dependen de él. Esta dependencia detiene el avance en este campo. Por ello, es necesario crear una metodología independiente del sistema y de la experiencia del auditor, reduciendo el esfuerzo necesario para llevar a cabo la labor de auditoría informática y, por consiguiente, sus costes de mantenimiento y formación.

Dentro del campo de la informática es posible crear un proceso que permita guardar y recuperar los datos de auditoria de base de datos de tal forma que sea capaz de procurar mantenerlos con un mínimo coste en infraestructura.

Los sistemas gestores de bases de datos cada vez aumentan los recursos en esta materia, y *Oracle* es el sistema gestor más avanzado del mercado actual. Su versión 11g ofrece herramientas de auditoria tanto genérica como de grano fino.

Sin embargo, un uso indebido de estas herramientas puede hacer que una base de datos se colapse o se caiga. Por ello era necesario estudiar estas herramientas para que pudiesen ser aprovechadas lo mejor posible.

Tras una labor de investigación y realización de pruebas, se ha pretendido estandarizar lo máximo posible la infraestructura y actividades necesarias para llevar a cabo la labor de auditoría informática con *Oracle* 11g.

Mediante una batería de pruebas, programas de medición de rendimiento y análisis de los datos se ha llegado a configurar una infraestructura de dos ordenadores que maximiza de forma general el rendimiento del servidor de base de datos de manera que se aumente la disponibilidad del mismo. El servidor contará con una base de datos *Oracle* 11g disponible para los usuarios, con un usuario auditor especial con permisos de consulta y borrado de tuplas de las tablas de auditoría. Las tablas de auditoría deberán estar en el mismo servidor, ya que de otra forma se podría colapsar la red y así bloquear la base de datos.

Para la realización de copia y backup de las tablas de auditoría así como su gestión de la configuración, se utiliza otro ordenador, que será usado por el auditor informático para realizar sus labores:

- Evolucionar las políticas de auditoría.
- Obtener información sobre hechos excepcionales.
- Obtener información sobre datos de auditoría.
- Realización de copias de seguridad y limpieza de las tablas de auditoría.

Todo ello cumpliendo con el objetivo fundamental del trabajo de investigación: maximizar la disponibilidad y eficiencia, tanto del ordenador servidor auditado como del ordenador en el que se tratan de analizar los datos de auditoría.

Esto no es suficiente. Es necesario también describir cómo debe el auditor realizar sus actividades. Por ello se han propuesto, mediante el análisis de los objetos disponibles en *Oracle* 11g para la recolección de los datos, un conjunto de procedimientos generales que abarcan las acciones que debe realizar el auditor para el correcto mantenimiento del sistema.

En este análisis se ha estudiado el uso del *Log Miner* como herramienta de auditoría, llegando a la conclusión de ser una herramienta muy pesada con respecto al uso de políticas de auditoría que ofrece *Oracle* 11g: auditoría genérica y de grano fino.

La búsqueda de un dato implica recuperar todas las copias de seguridad realizadas a las tablas de auditoría. A continuación se crearán vistas análogas a las que ofrece *Oracle* 11g llamadas *dba_audit_trail* y *dba_fga_audit_trail* para la mejor visualización de los datos. Mediante un conjunto de consultas y las vistas creadas, se podrá llegar a obtener los datos deseados.

Finalmente y para que cualquier base de datos pueda optar a un sistema de auditoria a su medida, se debe establecer unos niveles de auditoría en función de la seguridad requerida por la base de datos de la organización. Se ha llegado a la conclusión de que estos niveles son necesarios, ya que el rendimiento obtenido en la realización de las pruebas hacen ver que el coste de mantenimiento de la auditoría de base de datos varía con respecto al uso que se le de a la misma.

Los niveles de auditoría se han establecido en base a dos pilares imprescindibles que determinan la necesidad de auditoria de cualquier base de datos, desde una pequeña organización hasta una grande. Estos pilares son los siguientes:

- La ley: los diferentes niveles de infracciones establecidos en el artículo 44 de la LOPD [3].

- El sistema informático: la maximización de la disponibilidad y el rendimiento de la base de datos.

Con ello, se ha de tener tres niveles:

- Básico: abarca la protección de la base de datos para prevenir infracciones leves según el artículo 44.2 de la LOPD [3]. Se basa en establecer y documentar los permisos concedidos a los usuarios para detectar posibles huecos de seguridad.

- Medio: abarca la protección de la base de datos para prevenir infracciones graves según el artículo 44.3 de la LOPD [3]. Se basa en detectar posibles irregularidades en el funcionamiento de la base de datos.

- Alto: abarca la protección de la base de datos para prevenir infracciones muy graves según el artículo 44.4 de la LOPD [3]. Consiste en aumentar aún más la seguridad, sin disminuir la disponibilidad de la base de datos.

La organización deberá establecer el nivel de seguridad que cree más conveniente para su correcto funcionamiento y aplicarlo.

El área de la auditoría informática es un campo al que queda un largo camino que recorrer. Las bases de datos *Oracle* comenzaron a contar con cierta potencia en materia de auditoría muy recientemente, a partir de la versión 9i lanzada al mercado a mediados del año 2001y en la versión 10g, lanzada en el 2003. La capacidad de auditoría *Oracle* está limitada principalmente por el propio gestor. Debido a ello, cualquier producto comercial se encuentra a un paso por detrás, por lo que es necesario acudir a la experiencia del auditor.

Este trabajo pretende servir de base para el comienzo de un desarrollo de metodología que permita reducir costes y esfuerzo de mantenimiento y uso de la auditoría *Oracle*. Dicha metodología debe ser evolucionada, adaptándose a las

nuevas tecnologías que surjan y a las bases de datos distribuidas, cada vez más presentes.

Como se ha visto en el presente trabajo, la herramienta *Oracle Audit Vault* ha desarrollado un Datawarehouse que permite, en teoría, aumentar la eficacia de la recolección de los datos de auditoría. Sería muy necesario estudiar en profundidad esta herramienta con el objetivo de observar la efectividad del Datawarehouse y su rendimiento.

También habría que estudiar cómo guarda los datos de auditoría la herramienta *Intrust* de *Quest* y compararlo con *Oracle Audit Vault*. Así se podría ver si son herramientas que se quedan en interesantes, son útiles o llegan a ser imprescindibles, y muy importante, para qué organizaciones.

Además, sería muy positivo desarrollar y evolucionar un Datawarehouse que mejore la eficiencia de la recolección y recuperación de los datos de auditoría.

Como líneas futuras de investigación habrá que analizar cómo evoluciona el motor de *Oracle*, las máquinas y los sistemas operativos para así observar como incide la auditoría en Bases de Datos en el rendimiento.

Además existen herramientas comerciales, cuya evolución deberá ser seguida, que permiten diseñar alertas, que son procesos que detectan amenazas de seguridad estudiando irregularidades en el uso de la base de datos. Estas herramientas deben ser estudiadas con el fin de saber si esas alertas realmente funcionan y cómo funcionan.

Por otra parte, una vez obtenida la recolección de datos de auditoría tendremos que pasar a realizar análisis automáticos y manuales de la información recolectada. De tal forma de que el auditor pueda obtener de una forma amigable, e incluso visual, elementos generales de auditoría. Por ejemplo el número de

operaciones por usuario y por objeto, accesos por usuario o accesos por objeto y usuario, conexiones fallidas, operaciones DML o consultas fallidas, etc.

Pero no sólo debe quedar ahí. Se debe diseñar procesos para la búsqueda de patrones de conocimiento para que la propia herramienta encuentre comportamientos correctos o anómalos en el aplicativo. En este campo queda mucho por investigar todavía, y las herramientas de ingeniería están lejos de realizar estos procesos.

9. Referencias

[1] Piattini Velthuis, Mario G. Auditoría informática: un enfoque práctico. Editorial RA-MA, 1997

[2] PARRILLA GONZALEZ, Ana Belén. Auditoría de bases de datos. Editorial A.B Parrilla, 1997

[3] España. Ley orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal [en línea]. Boletín Oficial del Estado, 14 de diciembre de 1999, núm.298, p. 43088-43099. e http : // www.boe.es (consultado 25-01-09).

[4] Oracle Database SQL Reference 10g Release 2(10.2). Create Trigger [en línea]. [EE.UU]: Oracle, 2005 [ref. de 30 de noviembre de 2008]. Disponible en Web: <http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_7004.htm>

[5] Oracle 9i Database Daily Feature. Fine-Grained Auditing [en línea]. [EE.UU]: Oracle Technology Network [ref. de 7 de diciembre de 2008]. Disponible en web:

< http://www.oracle.com/technology/products/oracle9i/daily/oct03.html >

[6] Oracle Database Security Guide 10g Release 2(10.2). Configuring and Administering Auditing [en línea]. [EE.UU]: Oracle, 2008 [ref. de 8 de diciembre de 2008]. Disponible en Web:

<http://download.oracle.com/docs/cd/B19306_01/network.102/b14266/cfgaudit.htm>

[7] Oracle Database SQL Language Reference 11g Release 1(11.1) [en línea]. [EE.UU]: Oracle, September 2008 [ref. de 15 de diciembre de 2008]. Disponible en Web:

<http://download.oracle.com/docs/cd/B28359_01/server.111/b28286.pdf>

Cap.13. SQL Statements: ALTER TRIGGER to COMMIT: Audit.

[8] Oracle. E-Delivery [en línea]. [EE.UU]: Oracle, 2008 [ref. de 19 de diciembre de 2008]. Disponible en Web:

<http://edelivery.oracle.com/EPD/GetUserInfo/get_form?caller=LinuxWelcome>

[9] Oracle Technology Global Price List. Software Investment Guide [en línea]. [EE.UU]: Oracle, January 2008 [ref. de 23 de diciembre de 2008]. Disponible en Web:

<http://www.oracle.com/corporate/pricing/ePLext.PDF>

[10] Oracle Database 11g Release 1(11.1). About Database Storage Structures [en línea]. [EE.UU]: Oracle, 2008 [ref. de 7 de enero de 2009]. Disponible en Web:

<http://download.oracle.com/docs/cd/B28359_01/server.111/b28301/storage001.htm#ADMQS12053>

[11] Burleson Consulting. DBA_AUDIT_TRAIL view tips [en línea]. [EE.UU]: Burleson Enterprises, 2009 [ref. de 15 de enero de 2009]. Disponible en Web:

<http://www.praetoriate.com/data_dictionary/dd_dba_audit_trail.htm>

[12] Burleson Consulting. DBA_FGA_AUDIT_TRAIL view tips [en línea]. [EE.UU]: Burleson Enterprises, 2009 [ref. de 16 de enero de 2009]. Disponible en Web:

<http://www.praetoriate.com/data_dictionary/dd_dba_fga_audit_trail.htm>

[13] Mateos Macía, Enrique. Auditoría en bases de datos. Desarrollo e implementación con *Oracle*. Evaluación y comparativa. Universidad Carlos III de Madrid, 2007

[14] Burleson Consulting. DBA_AUDIT_TRAIL view tips [en línea]. [EE.UU]: Burleson Enterprises, 2009 [ref. de 28 de diciembre de 2008]. Disponible en Web:

<http://www.quest.com/intrust/>

[15] Oracle® Audit Vault Administrator's Guide [en línea]. [EE.UU]: Oracle, 2008 [ref. de 29 de diciembre de 2008]. Disponible en Web:

<http://download.oracle.com/docs/cd/E13850_01/doc.102/e13841/toc.htm>

Anexo I: elementos incluidos en el CD

script1.sql: muestra el ejemplo de alerta para el vigilar el espacio disponible en un espacio de tablas.

script2.sql: muestra la tabla usada en las pruebas.

script3.sql: muestra la sentencia de audiotoría genérica usada en las pruebas.

script4.sql: muestra las sentencias para crear politicas de auditoría de grano fino usadas en las pruebas.

script5.sql: muestra las sentencias para realizar copia y backup de AUD$.

script6.sql: muestra las sentencias para realizar copia y backup de FGA_LOG$.

script7.sql: fichero para la creación de la vista VISTA_AUDITORÍA.

script8.sql: fichero para la creación de la vista VISTA_AUDITORÍA_FGA.

script9.sql: ejemplos para obtener un conjunto de datos de auditoría genérica.

script10.sql: ejemplos para obtener un conjunto de datos de auditoría de grano fino.

prueba1-A.sql: script utilizado para realizar la prueba 1-A.

prueba1-B.sql: script utilizado para realizar la prueba 1-B.

prueba2.sql: script utilizado para realizar la prueba 2.

prueba3-A.sql: script utilizado para realizar la prueba 3-A.

prueba3-B.sql: script utilizado para realizar la prueba 3-B.

prueba4-A.sql: script utilizado para realizar la prueba 4-A.

prueba4-B.sql: script utilizado para realizar la prueba 4-B.

prueba5-A.sql: script utilizado para realizar la prueba 5-A.

prueba5-B.sql: script utilizado para realizar la prueba 5-B.

prueba6.sql: script utilizado para realizar la prueba 6.